Low Power Wireless Networked System Design

Rajesh K. Gupta

Center for Embedded Computer Systems

University of California, Irvine

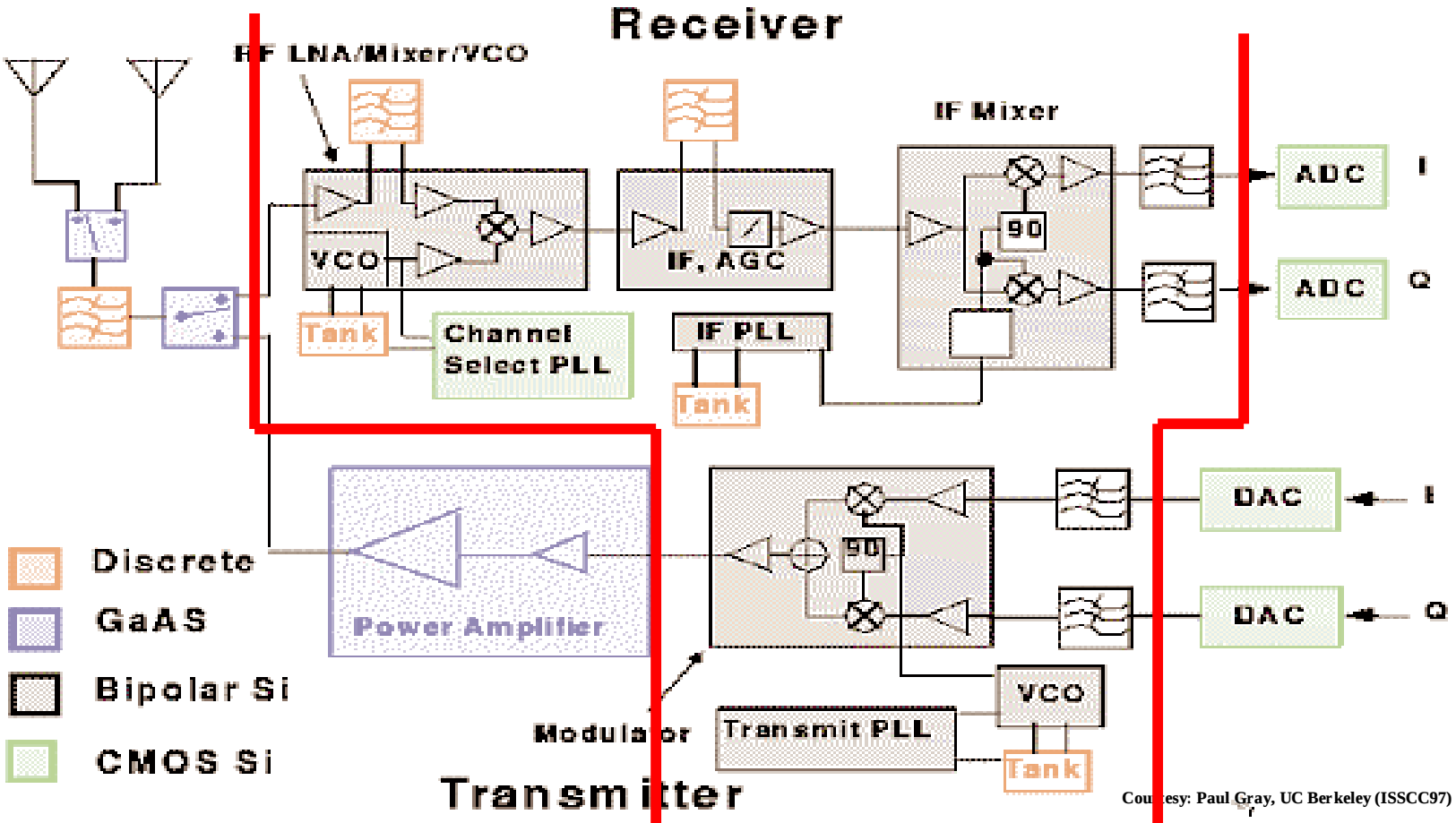
with contributions from

Mani Srivastava, UCLA, *mbs@ics.uci.edu.edu*

Nikil Dutt, UC Irvine, *dutt@ics.uci.edu*

HotChips August 18, 2002, Stanford, CA

RF Transceiver Block Diagram

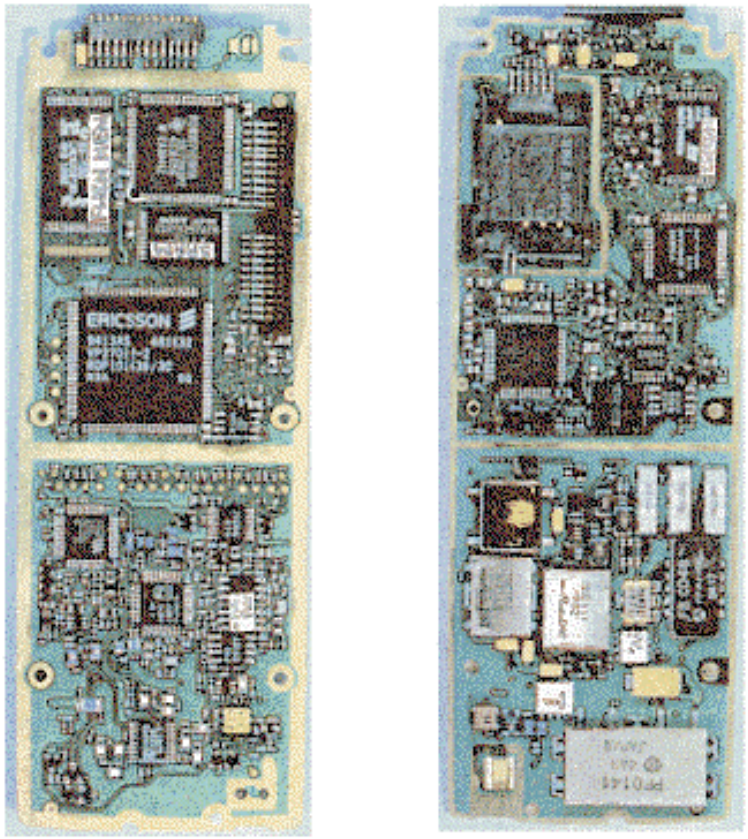


1. GaAs, Si Bipolar

2. Si Bipolar, BiCMOS

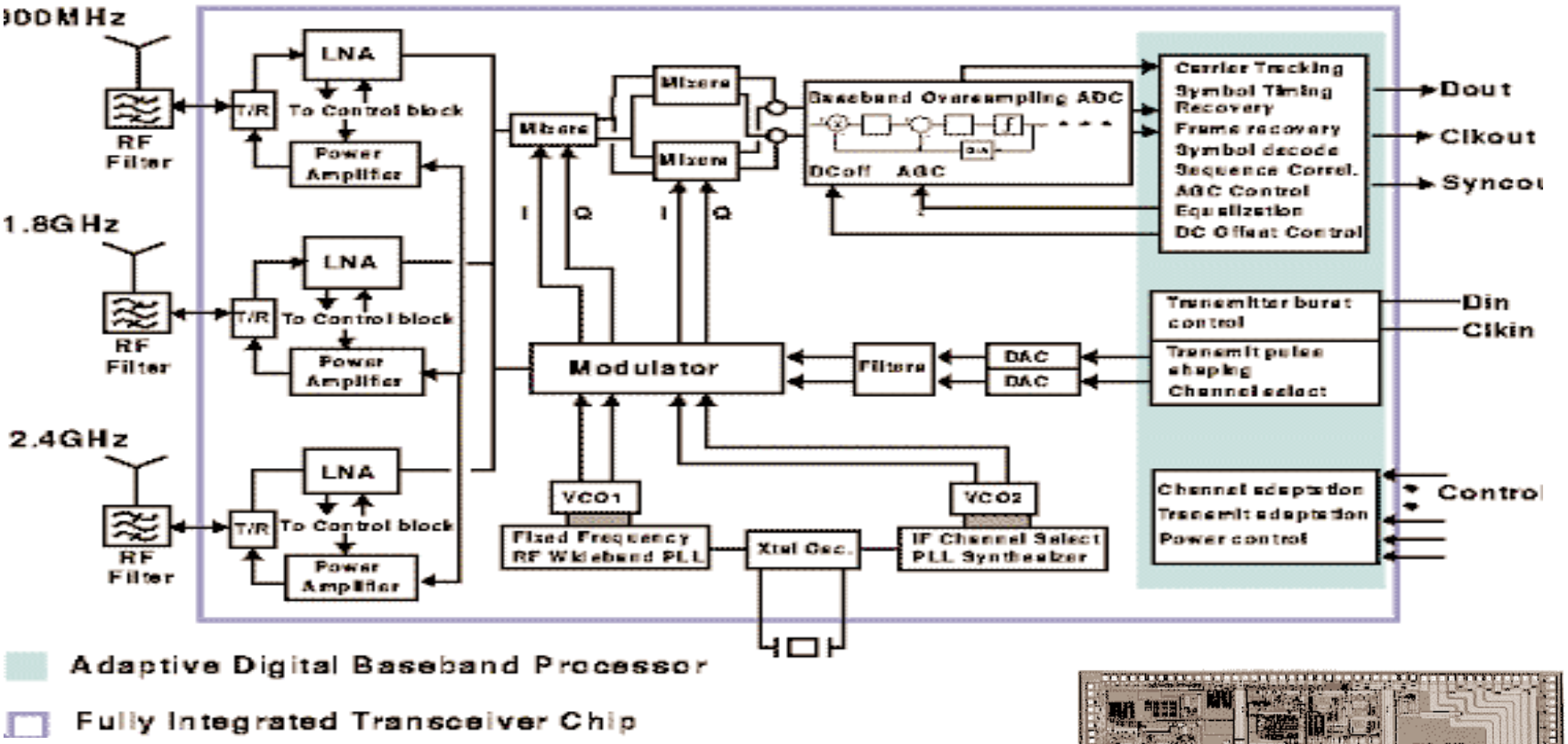
3. CMOS

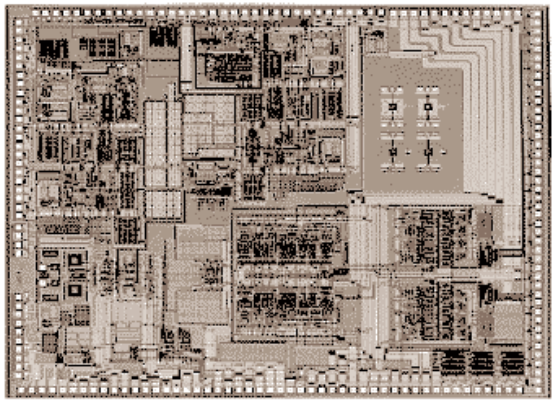
Typical RF Transceiver Implementation



Courtesy: Paul Gray, UC Berkeley (ISSCC97)

UCB DECT CMOS Adaptive RX





0.6 um CMOS, 7.5mm x 6.4mm die size.
55dB image rejection, mostly from IR mixer.
198 mW power. RX gain max 78 dB, min 26 dB

Courtesy: Rudell, et al, ISSCC'97

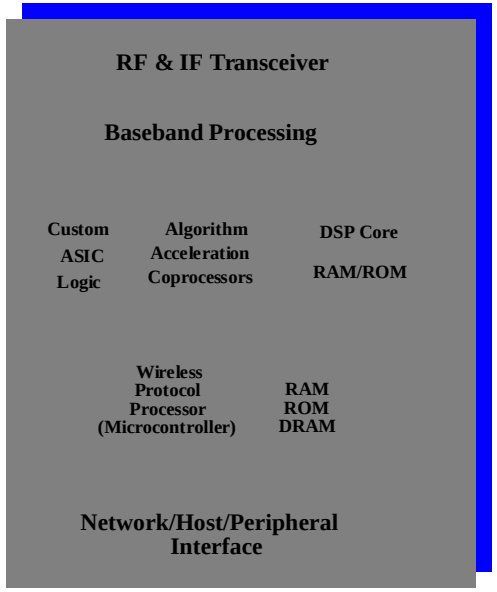
Wireless Systems Designs

- **Key Drivers**
  - **Relentless digitization of signals and systems**
    - **high speed digital circuits, ADCs leading to IF (and even RF) processing in digital domain, direct conversion techniques**
    - **complex communication algorithms and increasing available MIPS favor digital implementation**
  - **Microelectronic advances in CMOS VLSI**
    - **(RF as well as base-band digital processing), MEMS structures**

➡ **These trends are making it increasingly possible (and even cost effective) to build Wireless Systems on a Chip**

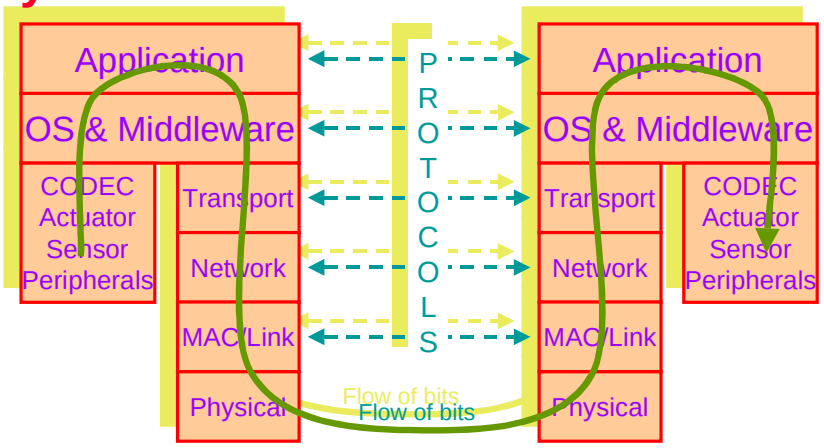
Networked Embedded Systems (NES)

- On-chip application computing
- On-chip communication and networking
- Indeed, complete integration of all layers of a networked node on a single chip
  - physical ⇒ transceiver, modem
  - link/MAC ⇒ packet scheduling
  - routing ⇒ routing protocols
  - transport ⇒ TCP
  - application ⇒ adaptive buffering

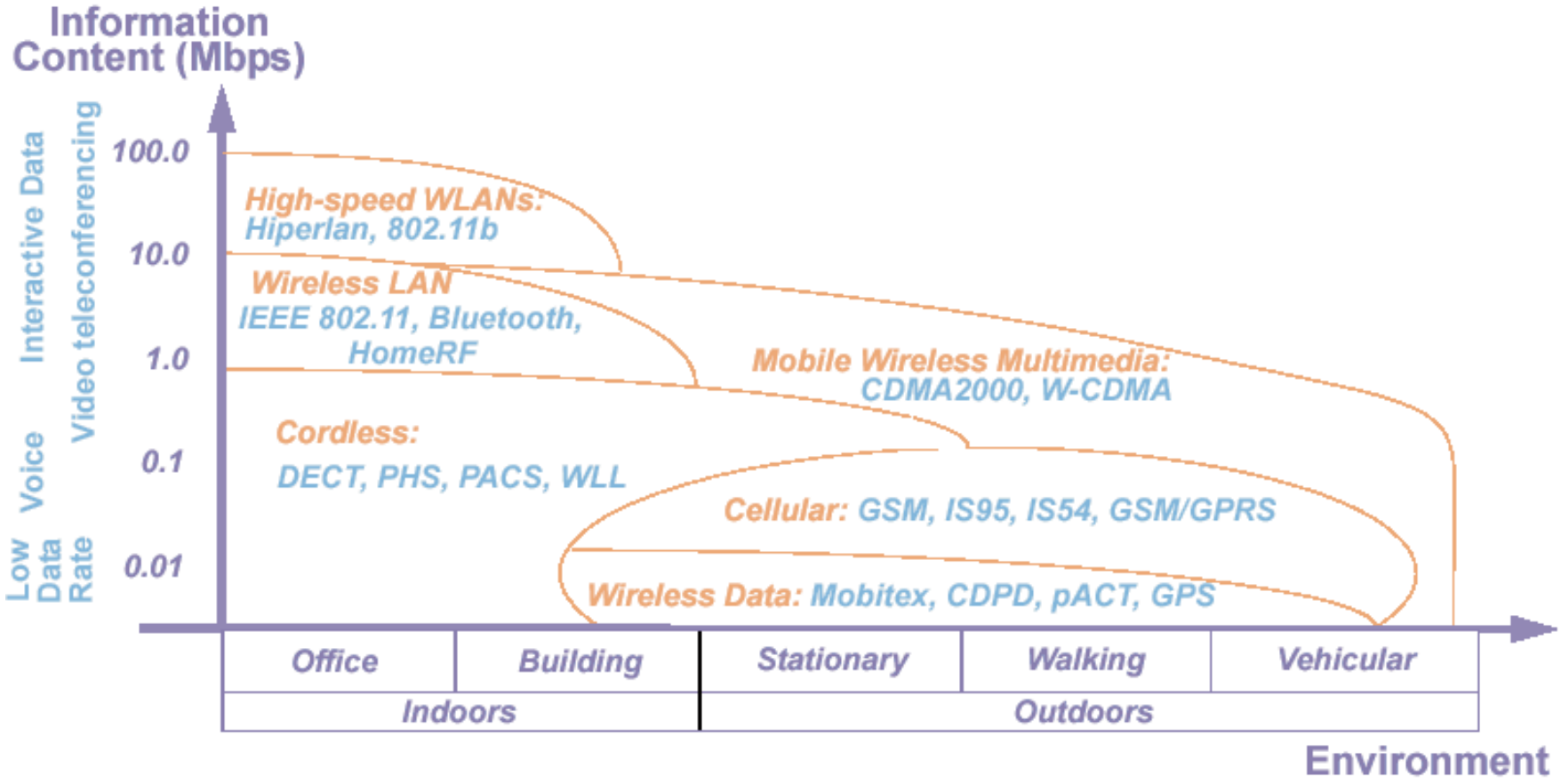


➡ IC system designer is also a networked system designer.

- Enormous application possibilities
  - Instrumented wide-area spaces
  - Personal area spaces
  - Internet end-points
  - In-body, In-cell, In-vitro (I3) spaces

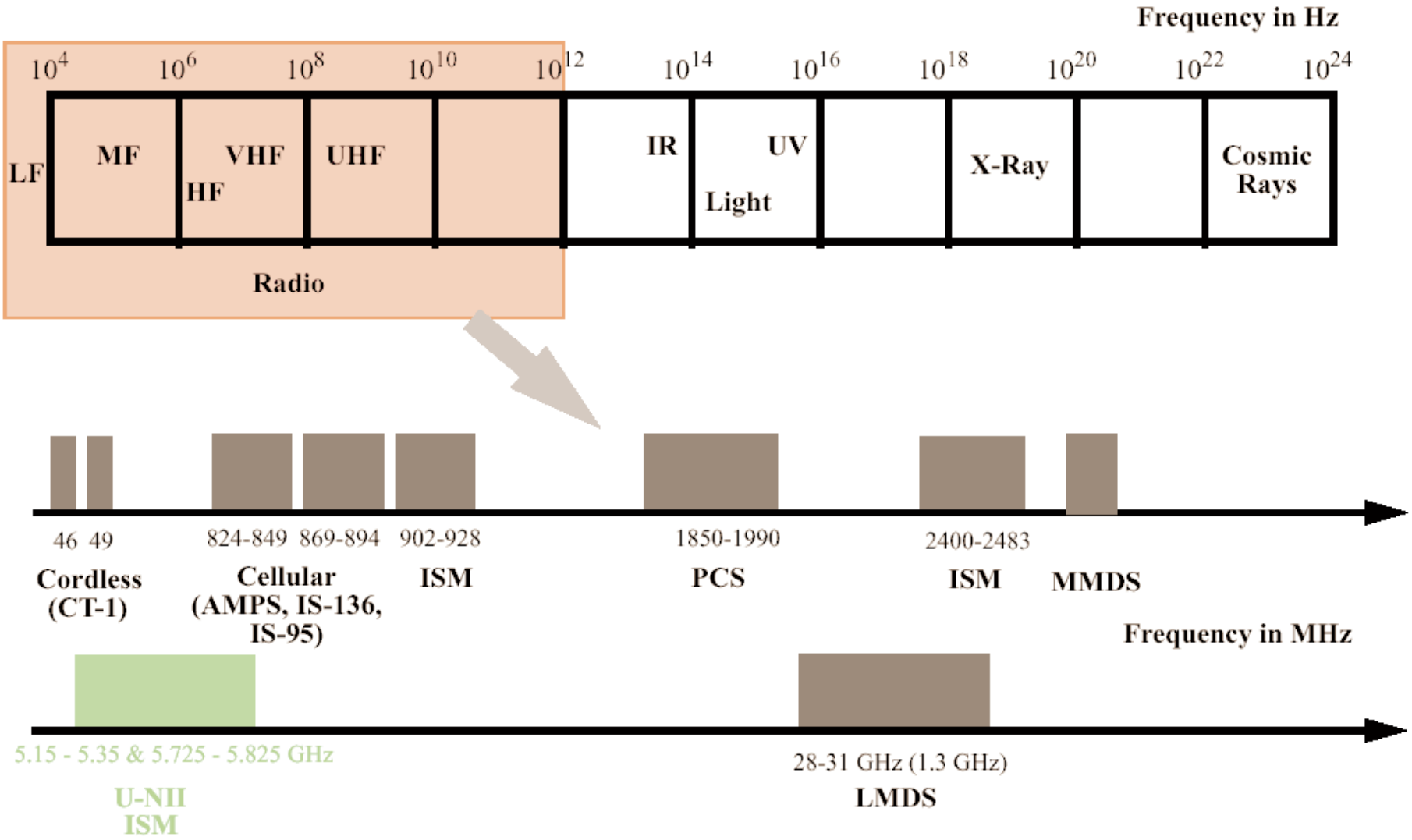


“Computers with Radios” are finding diverse applications

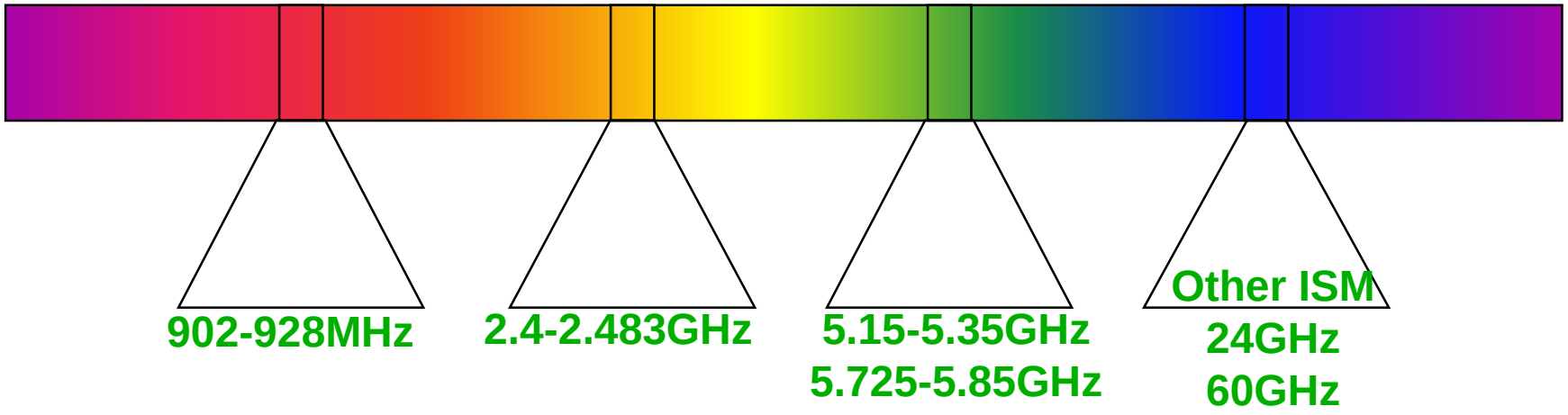


- **Networking over a wide range of environments.**

Seeking Valuable Spectrum Slots



Crowding in Unlicensed ISM Bands

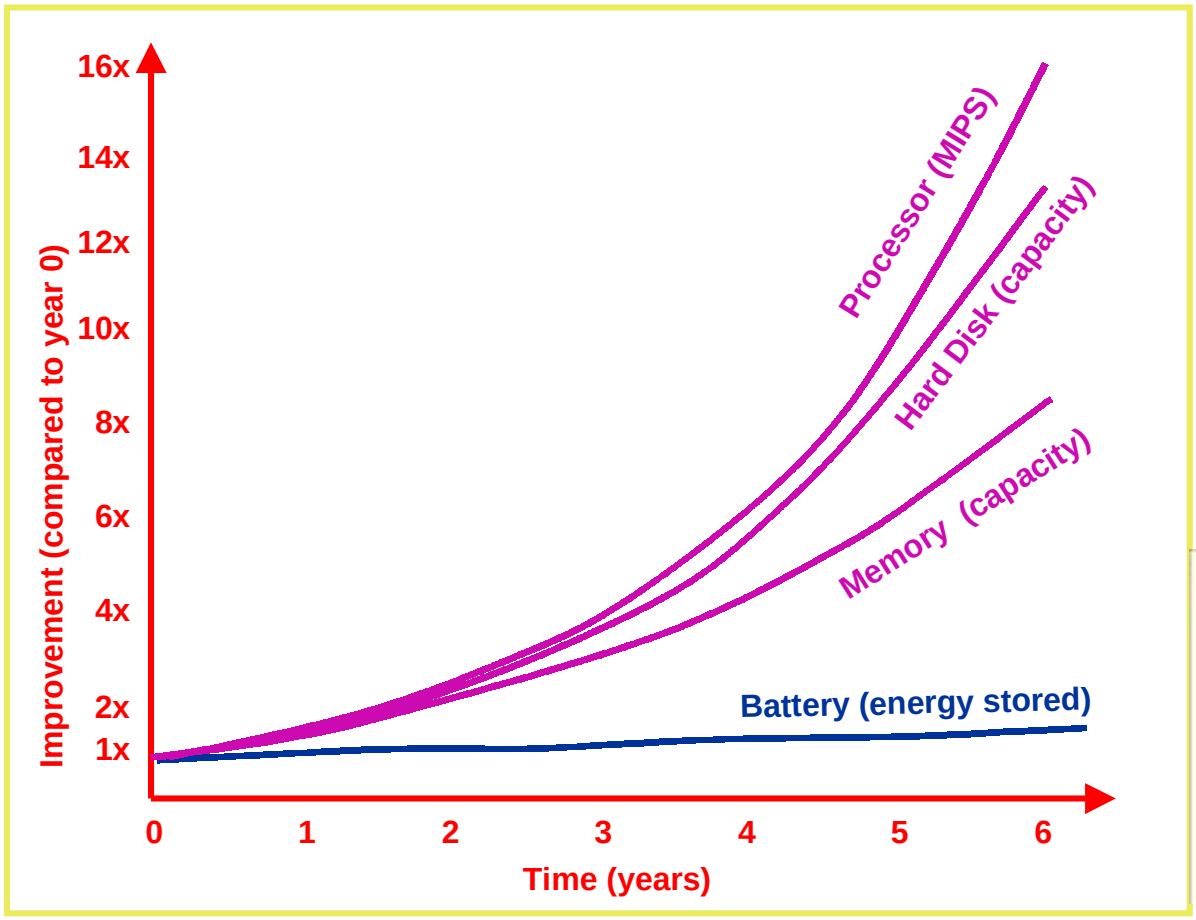


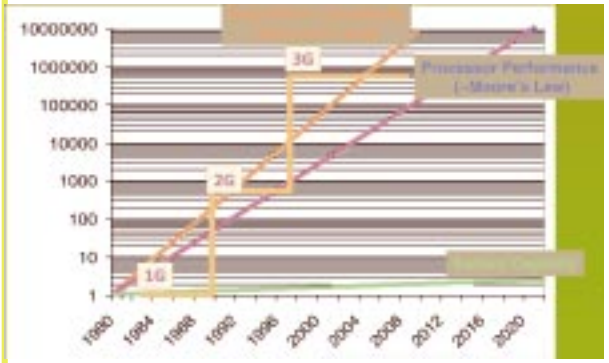
- 900MHz ISM band – cordless phones, some WAN, proprietary LAN, industrial heating
- 2.4GHz – cordless phones, Bluetooth™, 802.11b, HomeRF, microwave ovens
- 5GHz – mobile satellite, 802.11a, HiperLAN, HiperPAN, Cordless phones, 802.15.3 (proposed), microwave ovens (future), fixed wireless

Wireless NES System Characteristics

- **Wireless**
  - limited bandwidth, high latency (3ms-100ms)
  - variable link quality and link asymmetry due to noise, interference, disconnections
  - easier snooping
  - need for more signal and protocol processing
- **Mobility**
  - causes variability in system design parameters: connectivity, b/w, security domains, location awareness
  - need for more protocol processing
- **Portability**
  - limited capacities (battery, CPU, I/O, storage, dimensions)
  - need for energy efficient signal and protocol processing

Energy Availability Growth
limited to 2-3% per year





J. Rabaey, BWRC

➨ **Need to be energy efficient at all levels and in all tasks.**

Computational Efficiency

- **Speed power efficiency has indeed gone up**
  - **10x / 2.5 years for µPs and DSPs in 1990s**
    - **between 100 mW/MIP to 1 mW/MIP since 1990**
  - **IC processes have provided 10x / 8 years since 1965**
  - **rest from power conscious IC design in recent years**
- **Lower power for a given function & performance**
  - **e.g. 1.6x / year reduction since early 80s for DSPs (source TI)**
- **Most optimistic projections at best stop at 60 pJ/op (about 20X)**

Processor	MHz	Year	SPECint-95	Watts
P54VRT (Mobile)	150	1996	4.6	3.8
P55VRT (Mobile MMX)	233	1997	7.1	3.9
PowerPC 603e	300	1997	7.4	3.5
PowerPC 604e	350	1997	14.6	8
PowerPC 740 (G3)	300	1998	12.2	3.4
PowerPC 750 (G3)	300	1998	14	3.4
Mobile Celeron	333	1999	13.1	8.6

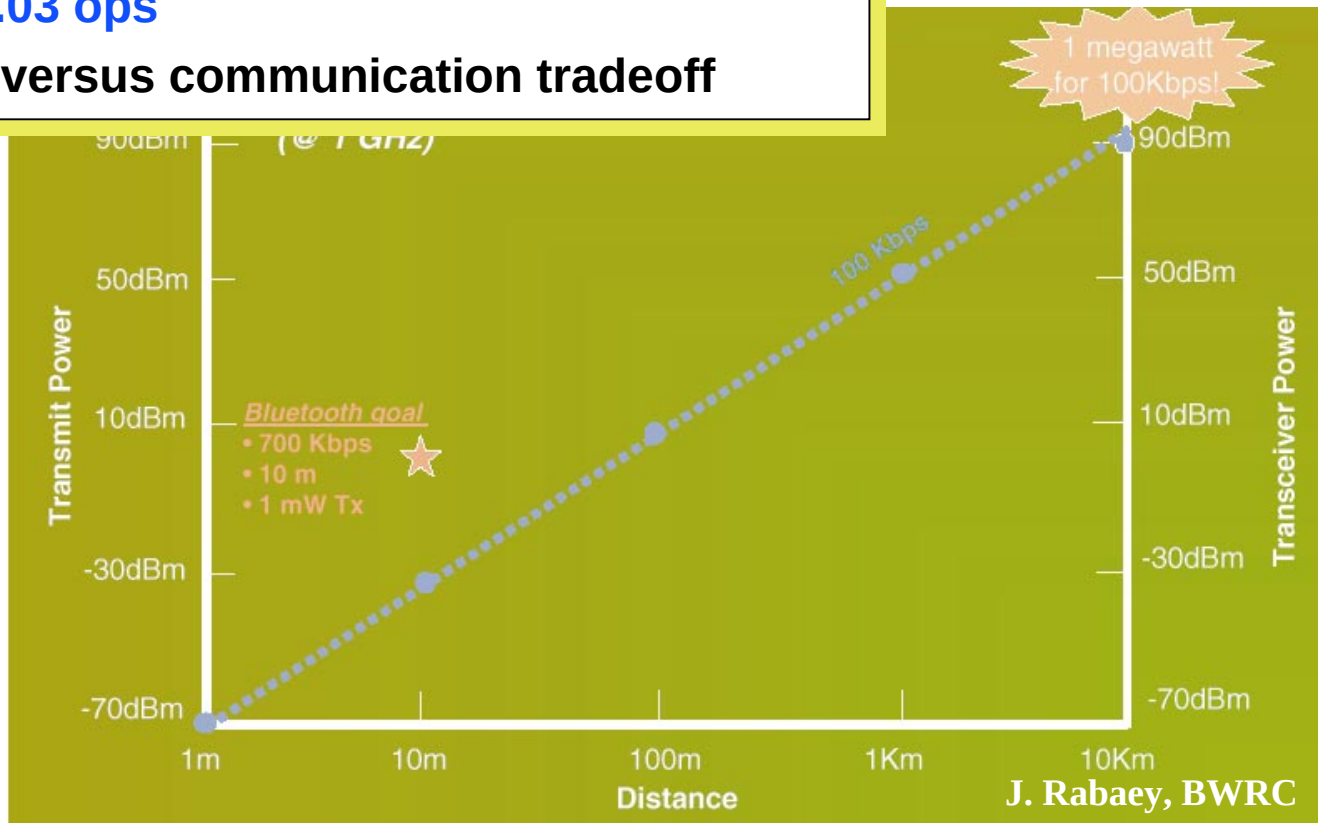
- **However, circuit gains are nearing a plateau**
  - **circuit tricks & voltage scaling provided a large part of the gains**
- **while energy needs (functionality, speed) continue to climb**
  - **10x increases: in gate count (7 years); in frequency (9 years)**

Efficiency in Communications

- Power Efficiency (or Energy Efficiency) $\eta_P = E_b/N_0$
  - ratio of signal energy per bit to noise power spectral density required at the receiver for a certain BER
  - high power efficiency does not mean low power
- Bandwidth Efficiency η_B = bit rate / bandwidth = R_b/B bps/hz
  - ratio of throughput data rate to bandwidth occupied by the modulated signal
- Often a trade-off between the two
  - e.g. for a given BER
    - adding FEC reduces η_B but reduces required η_P
    - modulation schemes with larger # of bits per symbol have higher η_B but also require higher η_P

Communication vs. Computation

- Computation cost (2004 projected): 60 pJ/op
- Minimum thermal energy for communications:
  - 20 nJ/bit @ 1.5 GHz for 100 m
    - equivalent of 300 ops
  - 2 nJ/bit @ 1.5 GHz for 10 m
    - equivalent of 0.03 ops
- ➡ significant processing versus communication tradeoff



The Need

- Power consumption, energy efficiency is a system level design concern
  - efficiency in computation, communication and networking subsystems
- The energy/power tradeoffs cut across
  - all system layers: circuit, architecture, software, algorithms
  - need to choose the right metric
- Power awareness goes beyond low power concerns
  - make tradeoffs against performance, quality measures against application constraints

Tutorial Focus

Goal: A systems view of integrated wireless system design for low power.

- System-level power management
  - architectural, networking, software approaches
  - will not cover process, device, circuits and VLSI for low power
- "Management" of power and energy
  - techniques that achieve specific functionality, performance goals within available energy (and dynamically changing) power, energy constraints
- Targeted at "on-chip computing and networked systems"
  - systems with interesting network interfaces, computation and communication capabilities.

This tutorial will NOT describe:

- detailed algorithms for dynamic power management
- theory of radio and communications system design
- detailed architecture of any wireless communication system.

Outline

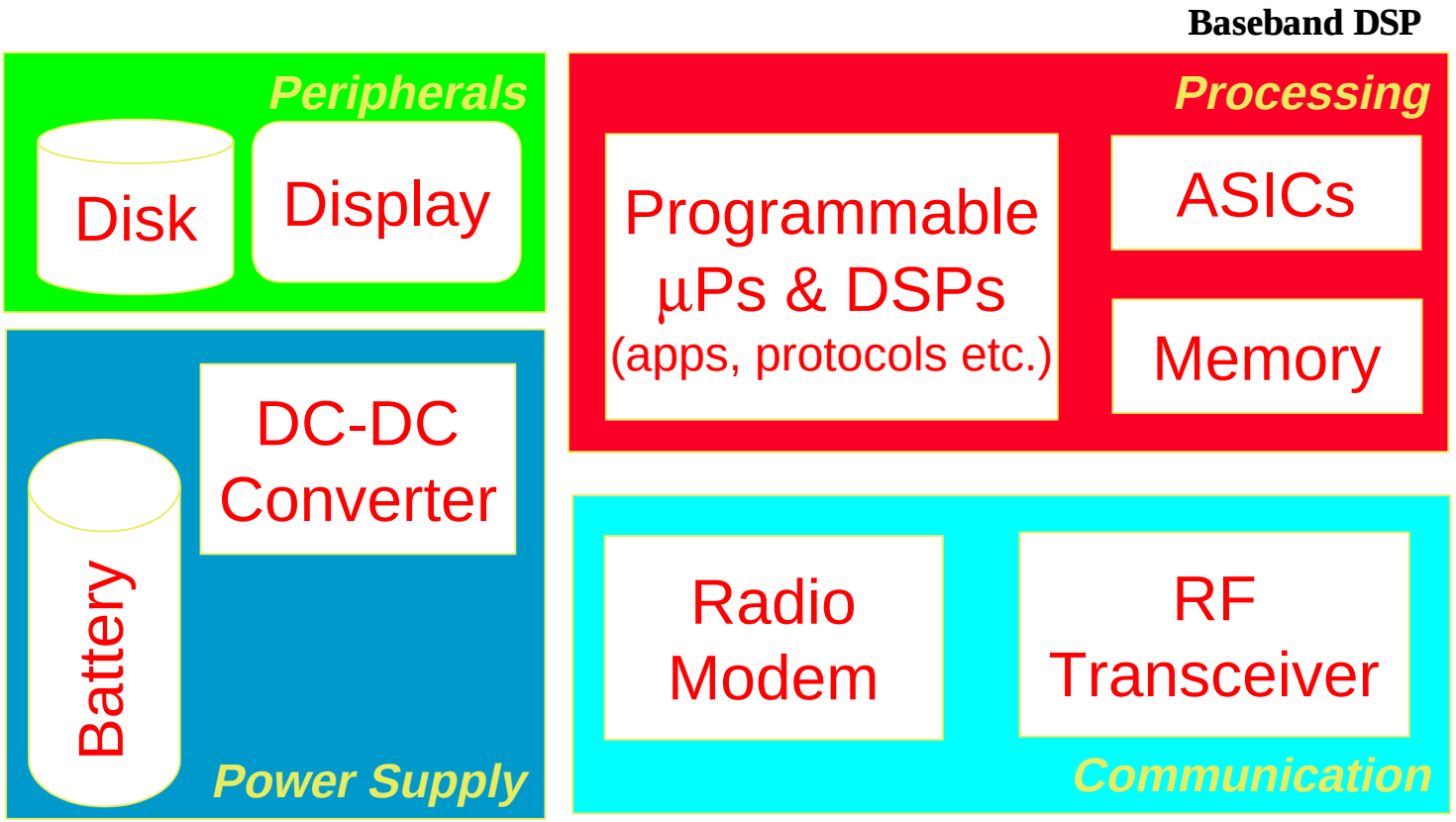
- Part I: Introduction to networked embedded systems
  - Energy consumption characteristics in NES
  - Power metrics
- Part II: Power management strategies
  - Effect of battery
  - Slowdown versus shutdown in DPM
  - Communications and networking strategies
  - Architectural, CPU and software strategies
  - OS strategies
- Part III: Real Life Examples
  - Processors
  - Protocol standards
- Summary

Portability in NES

- Networked SOCs are finding use far beyond traditional desktop machines
  - Pocket computers, PDAs, wireless pads, wireless sensors, pagers, cell phones
  - Efficient power use is crucial to portability, reliability and thermal management
- Energy and power usage of these devices is markedly different from laptop and notebook computers
  - much wider dynamic range of power demand
  - increasing share of memory, communication and signal processing subsystems (as opposed to disk storage, displays)
  - multiple power use modalities depending upon application:
    - "immortal", "paging-mode RX", "lifeline TX", "mission mode"

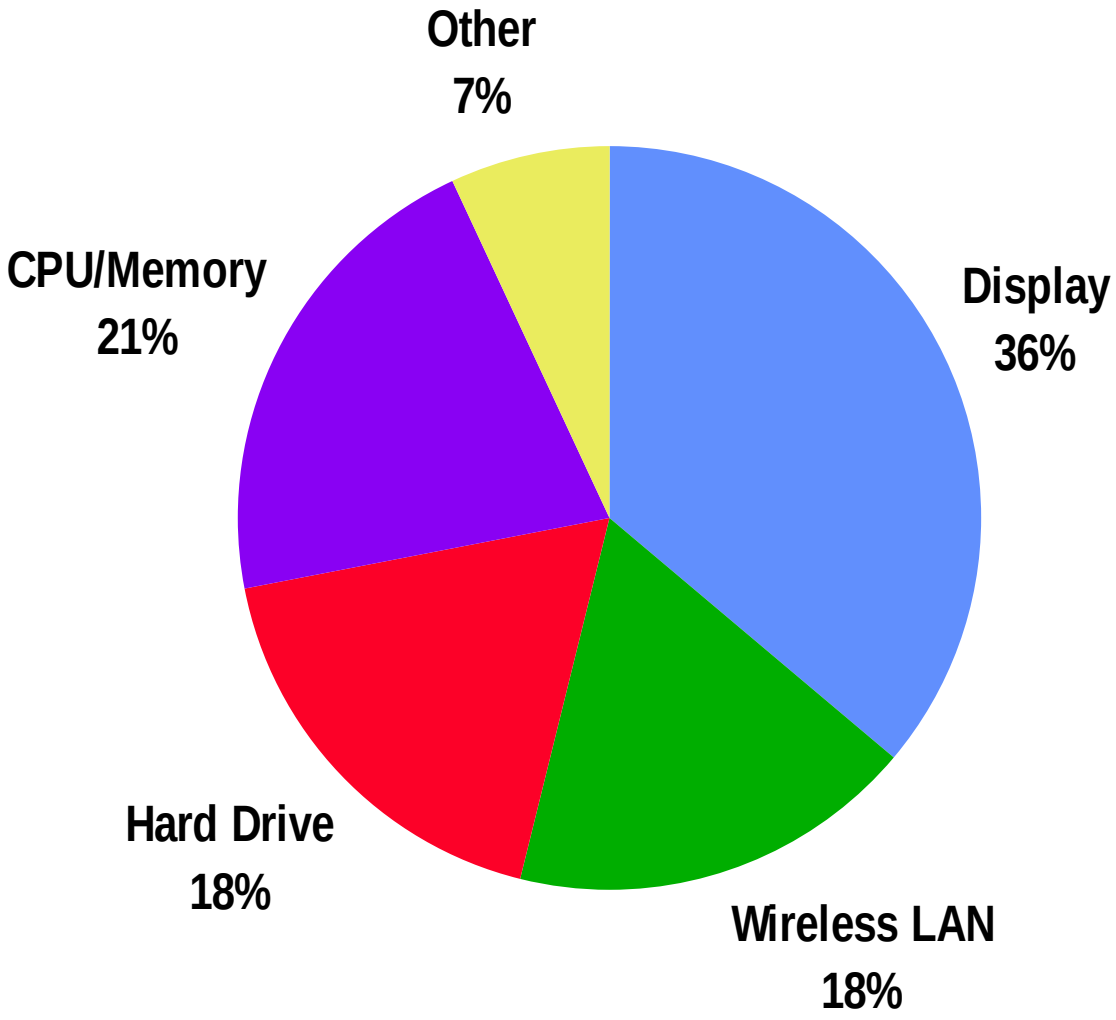
➡ Design of power-aware higher layer applications & protocols

Where does the Power Go?



Signaling protocols, choice of modulation, TX/RX architecture, RF/IF circuits

Example: Power Consumption for a Computer with Wireless NIC



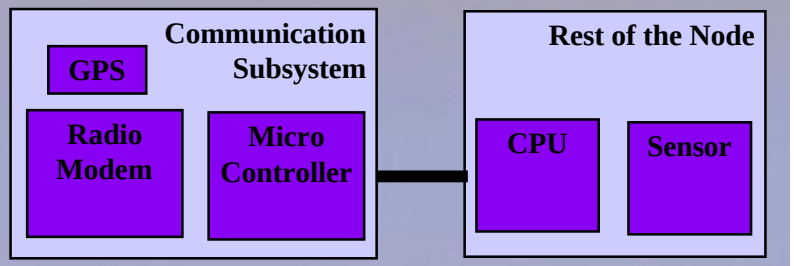
Example 1: Power Measurements on Rockwell WINS Node

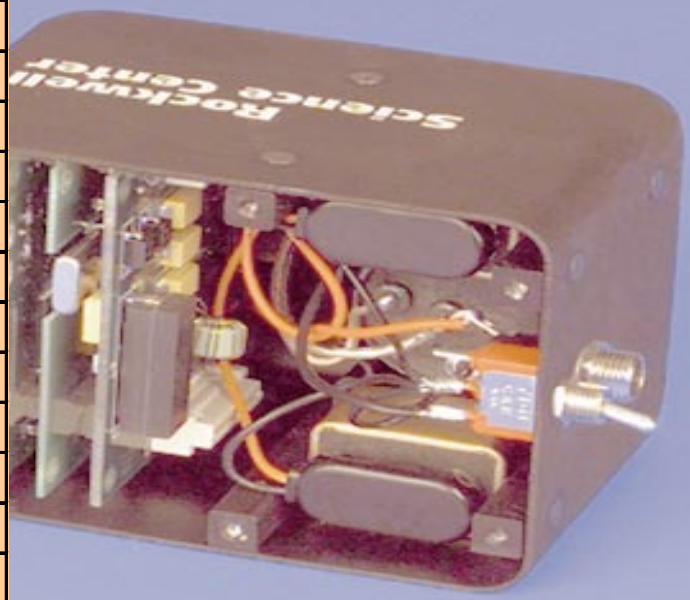
Processor	Seismic Sensor	Radio	Power (mW)
Active	On	Rx	751.6
Active	On	Idle	727.5
Active	On	Sleep	416.3
Active	On	Removed	383.3
Active	Removed	Removed	360.0
Active	On	Tx (36.3 mW)	1080.5
		Tx (27.5 mW)	1033.3
		Tx (19.1 mW)	986.0
		Tx (13.8 mW)	942.6
		Tx (10.0 mW)	910.9
		Tx (3.47 mW)	815.5
		Tx (2.51 mW)	807.5
		Tx (1.78 mW)	799.5
		Tx (1.32 mW)	791.5
		Tx (0.955 mW)	787.5
		Tx (0.437 mW)	775.5
		Tx (0.302 mW)	773.9
		Tx (0.229 mW)	772.7
		Tx (0.158 mW)	771.5
		Tx (0.117 mW)	771.1

Summary

- **Processor = 360 mW**
  - **doing repeated transmit/receive**
- **Sensor = 23 mW**
- **Processor : Tx = 1 : 2**
- **Processor : Rx = 1 : 1**
- **Total Tx : Rx = 4 : 3 at maximum range**

Capabilities: vibration, acoustic, accelerometer, magnetometer, temperature sensing



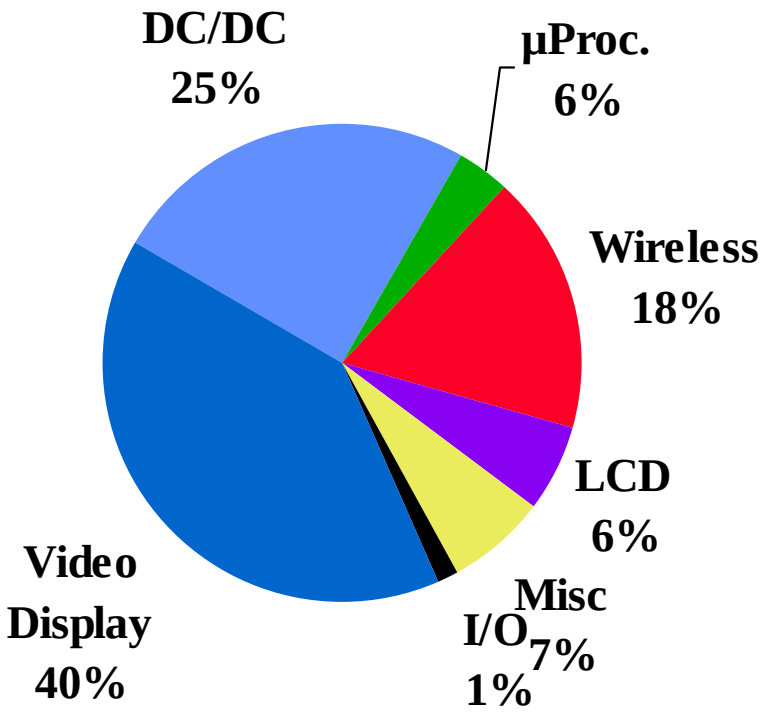


Example 2: Power Consumption for Compaq WRL's Itsy Computer

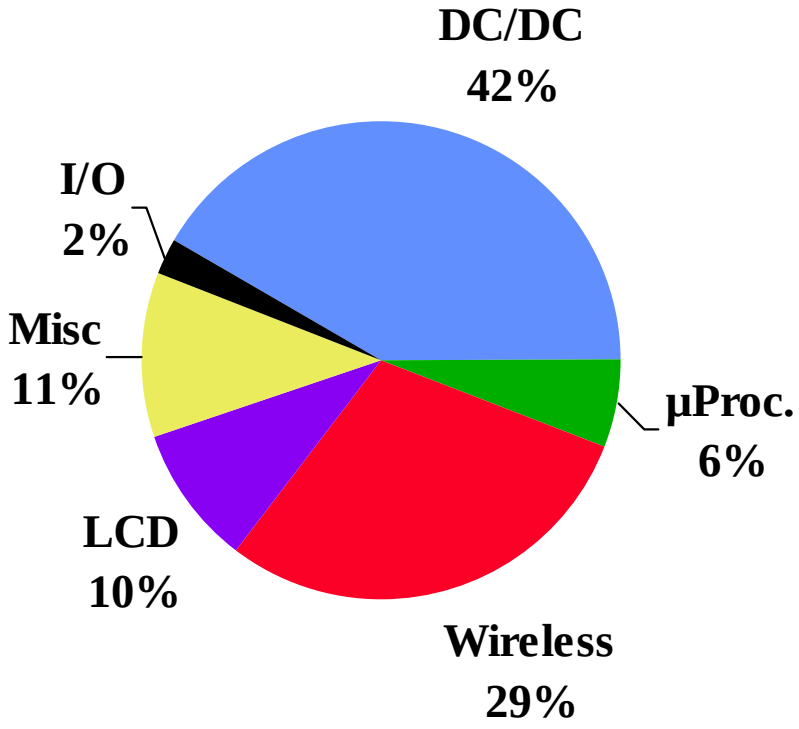
Itsy v1
StrongARM 1100
59–206 MHz (300 us to switch)
2 core voltages (1.5V, 1.23V)
64M DRAM / 32M FLASH
Touchscreen & 320x200 LCD
codec, microphone & speaker
serial, IrDA

- System power < 1W
  - doing nothing (processor 95% idle)
    - 107 mW @ 206 MHz
    - 77 mW @ 59 MHz
    - 62 mW @ 59 MHz, low voltage
  - MPEG-1 with audio
    - 850 mW @ 206 MHz (16% idle)
  - Dictation
    - 775 mW @ 206 MHz (< 0.5% idle)
  - text-to-speech
    - 420 mW @ 206 MHz (53% idle)
    - 365 mW @ 74 MHz, low voltage (< 0.5% idle)
- Processor: 200 mW
  - 42-50% of typical total
- LCD: 30-38 mW
  - 15% of typical total
    - 30-40% in notebooks

Example 3: Power Consumption for Berkeley's InfoPad Terminal



With Optional Video Display
Total = 9.6W
(with processor at 7% duty cycle)



Without Optional Video Display
Total = 6.8W
(with processor at 7% duty cycle)

Metrics for Power

- Absolute power (mW)
  - sets battery life in hours
  - problem: power ∝ frequency (slow the system!)
- uW/MHz
  - average energy consumed by the system
- Energy per operation
  - fixes obvious problem with the power metric
  - but can cheat by doing stuff that will slow the chip
    - ➤Energy/op = Power * Delay/op
- Metric should capture both energy and performance: e.g. Energy/Op * Delay/Op
  - Energy*Delay = Power*(Delay/Op)^2
- Therefore:
  - uW/MIPS: average energy per instruction
  - uW/MIPS^2: normalizes uW/MIPS with the architectural performance
    - useful for comparing architectures for power efficiency.

The Et^2 Metric

- Proposed as a metric for time and energy efficiency of computation by Alain Martin, CalTech
  - E = dynamic energy dissipated by a computation
  - t = latency or the cycle time of the computation
- Suitable in comparing CMOS VLSI circuits operating outside velocity saturation (i.e., when energy and delay can be adjusted by changing the supply voltage and Vt scaling)
- Et^2 is independent of supply voltage
- When multiple adjustable supply voltages are available, optimizing for Et^2 implies optimal energy and delay
  - thus, the metric can be used for transistor sizing

8-bit comparator	E (10pJ)	t (ns)	Et	Et^2
Linear (3.3)	25.24	3.93	99.21	389.97
Log (3.3)	44.97	2.35	105.50	247.57
Log (2.15)	16.52	3.93	64.97	255.59

Evaluating A Power Management Strategy

- Know the power bottlenecks in the system
  - effect on system vs. component power consumption
- Be fair
  - compare to current, and not the worst, strategy!
- Maximize work done, and not battery life
  - e.g. impact of system slowdown on user efficiency
- Consider the effect on other components
  - e.g. disk and display will stay on longer if CPU speed is cut in half
- Battery capacity is not a constant
  - depends on the power consumption level and profile.

Effectiveness of Power Management Algorithms

- A Model
  - Functionality is composed of individual tasks
  - Each task dissipates power to service requests that arrive over time
  - Inter-arrival time of requests is unknown
  - Requests are of different sizes and must be served in the order received
  - Task can choose to move to power minimizing states
- DPM is an on-line problem
  - input sequence is received dynamically during run-time
  - characteristics of the input sequence is not known
  - any algorithm to solve the problem cannot make static decisions about the input
- Competitive analysis provides a framework for understanding online strategies

Competitive Analysis

- Strategy S has a Competitive Ratio, *r*
  - if for all input sequence, σ, $C_S(\sigma) <= r \,.\, C_{opt}(\sigma)$
- Think of a 2-player game against a malicious adversary
  - Adversary creates an input sequence dynamically
  - Adversary knows the strategy, and creates inputs that makes the strategy perform as non-optimally as possible
  - Adversary is aware of the move of the strategy against every input
- Competitive Ratio akin to Complexity Lower Bounds
  - Represents the worst possible scenario
  - The adversary can be automatically generated using model checking approaches

Node Level Power Management

- Choices: H/W, Firmware, OS, Application, User

- Hardware & firmware
  - don't know the global state and application-specific knowledge
- Users
  - don't know component characteristics, and can't make frequent decisions
- Applications
  - operate independently
  - and the OS hides machine information from them
- OS is the most reasonable place, but…
  - OS should incorporate application information in power management
  - OS should expose power state and events to applications for them to adapt.

Power Consumption in Wireless NES

- There are two components of power
  - Instantaneous Power Consumption
    - directly affected by transceiver architecture and RF circuit design
    - a wide variation in power efficiency of RF front-ends
      - ➤paging receivers (930MHz carrier) with 1uV signal detection can last for months on a single AAA cell
      - ➤cell phones with essentially similar sensitivity characteristics are about 10X worse.
    - Not covered here.
  - Average Power Consumption
    - affected by communication protocols and power management strategies

➨ once you are done with all the low power tricks, "duty cycling" remains the most effective strategy to reduce power.

Outline

- Part I: Introduction to networked embedded systems
  - Energy consumption characteristics in NES
  - Power metrics
- **Part II: Power management strategies**
  - **Effect of battery**
  - **Slowdown versus shutdown in DPM**
  - **Communications and networking strategies**
  - **Architectural, CPU and software strategies**
  - **OS strategies**
- Part III: Real Life Examples
  - Processors
  - Protocol standards
- Summary

Battery Characteristics

- Focus on extending battery life
  - almost all other metrics have some loophole or the other
- Important battery characteristics from battery chemistry
  - energy density (Wh/liter), specific energy (Wh/kg)
  - power density (W/liter), specific power (W/kg)
  - open-circuit voltage, operating voltage
  - cut-off voltage, shelf life (due to leakage), cycle life

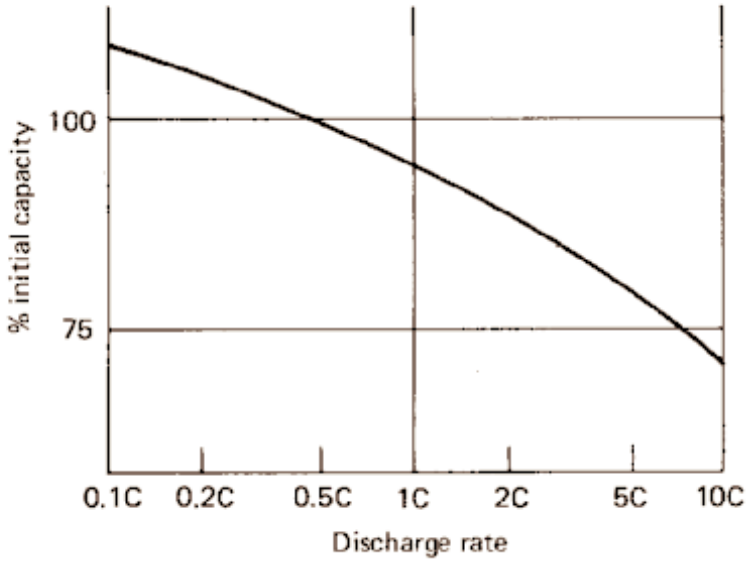
Battery	Rechargeable?	Wh/lb	Wh/litre
Alkaline MnO2	NO	65.8	347
Silver Oxide	NO	60	500
Li/MnO2	NO	105	550
Zinc Air	NO	140	1150
NiCd	YES	23	125
Li-Polymer	YES	65-90	300-415

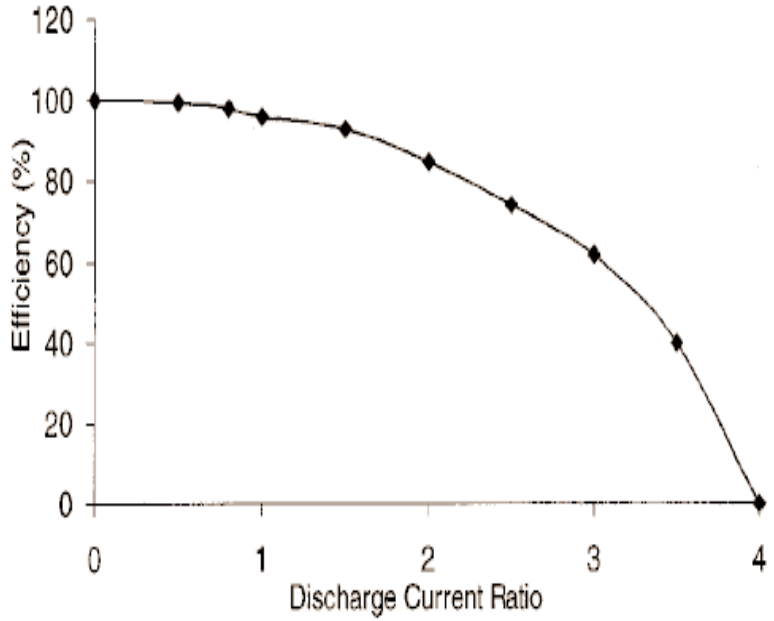
Battery Capacity

- Real-life battery capacity depends upon
  - discharge rate or the current drain
  - discharge profile and duty cycle
  - operating voltage and power level at which drained
- "C" rating for current is load current normalized to battery capacity
  - e.g., a discharge current of 1C for a capacity of 500 mAh is 500 mA
  - Peukert's formula: Energy capacity, $C = k / I^{\alpha}$ where α is upto 0.7 for most loads.

Battery Characteristics

- Manufacturer's often given battery efficiency (%) versus discharge rate (= I_avg / I_rated)
  - efficiency determines actual depletion in stored energy
  - e.g., for a 60% battery efficiency with rated capacity 100 mAh at 1V
    - computed average DC-DC current of 300 mA would drain the battery in 12 minutes, while at 100% efficiency it would last for 20 minutes
  - efficiency generally decreases as discharge rate increases.





Battery Discharge Profile

- Battery capacity can be extended by a “pulsed” discharge profile
  - the longer the battery rest period, the better is the recovery in battery capacity
  - yields higher specific power for a given specific energy or higher specific energy for a given specific power
- Battery-aware design
  - gain in battery life possible if system shutdown is done taking into account pulsed battery discharge
    - CPU shutdowns, slowdown
    - peripheral shutdown
    - protocols that adjust idle period based on the differential between TX, RX, Idle power

CMOS Digital Logic

- There are three components to power dissipation
  - Dynamic power consumption
    - charging and discharging capacitors
  - Short circuit currents
    - short circuit path between supply rails during switching
  - Leakage current from reverse biased junctions

$$P = A.C.V^2.f + A.I_{sw}.V.f + I_{leak}.V$$

where

A = activity factor (probability of $0 \rightarrow 1$ transition)

C = total chip capacitance

V = total voltage swing, usually near the power supply voltage

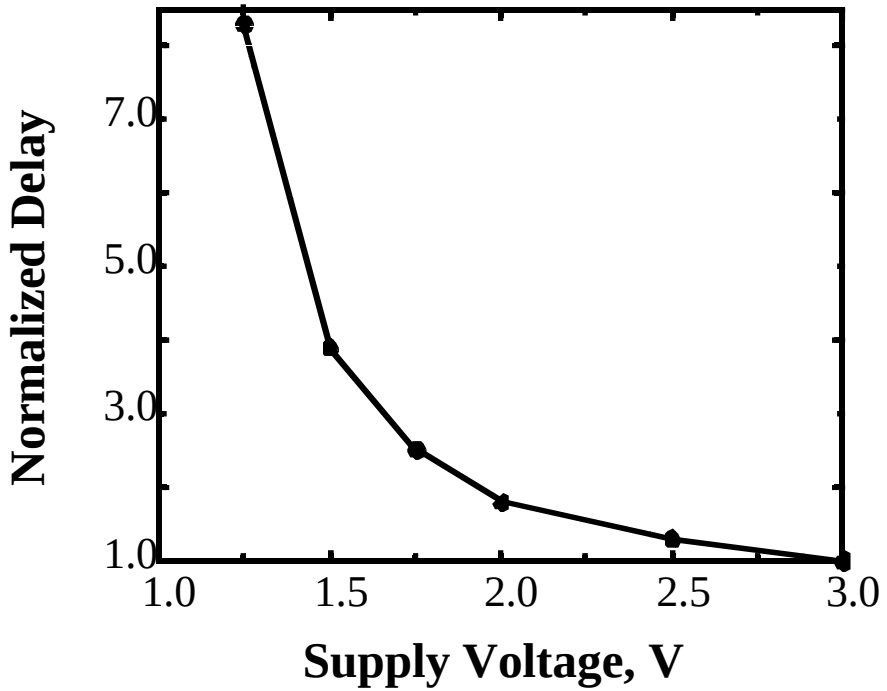
f = clock frequency

I_{sw} = short circuit current when logic level changes

I_{leak} = leakage current in diodes and transistors

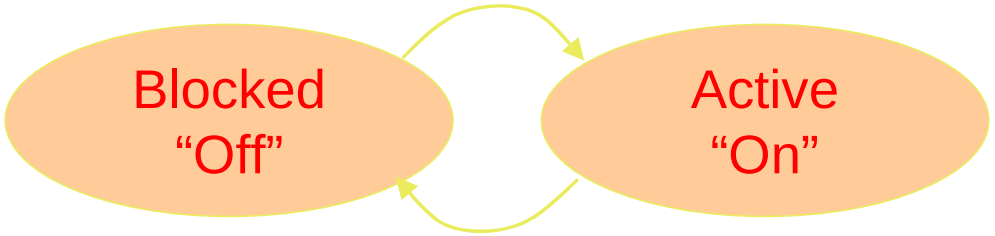
Slowdown by reducing supply voltage

- Reduction in supply voltage reduces speed
- Reduce supply voltage when
  - slower speed can be tolerated
  - or use architectural techniques to combat slow operation
    - e.g. concurrency, pipelining via compiler techniques



Shutdown for Energy Saving

- **Shutdown attractive for many wireless applications due to low duty cycle of many subsystems:**



T_{block} T_{active} **ideal improvement = 1 + T_{block}/T_{active}**

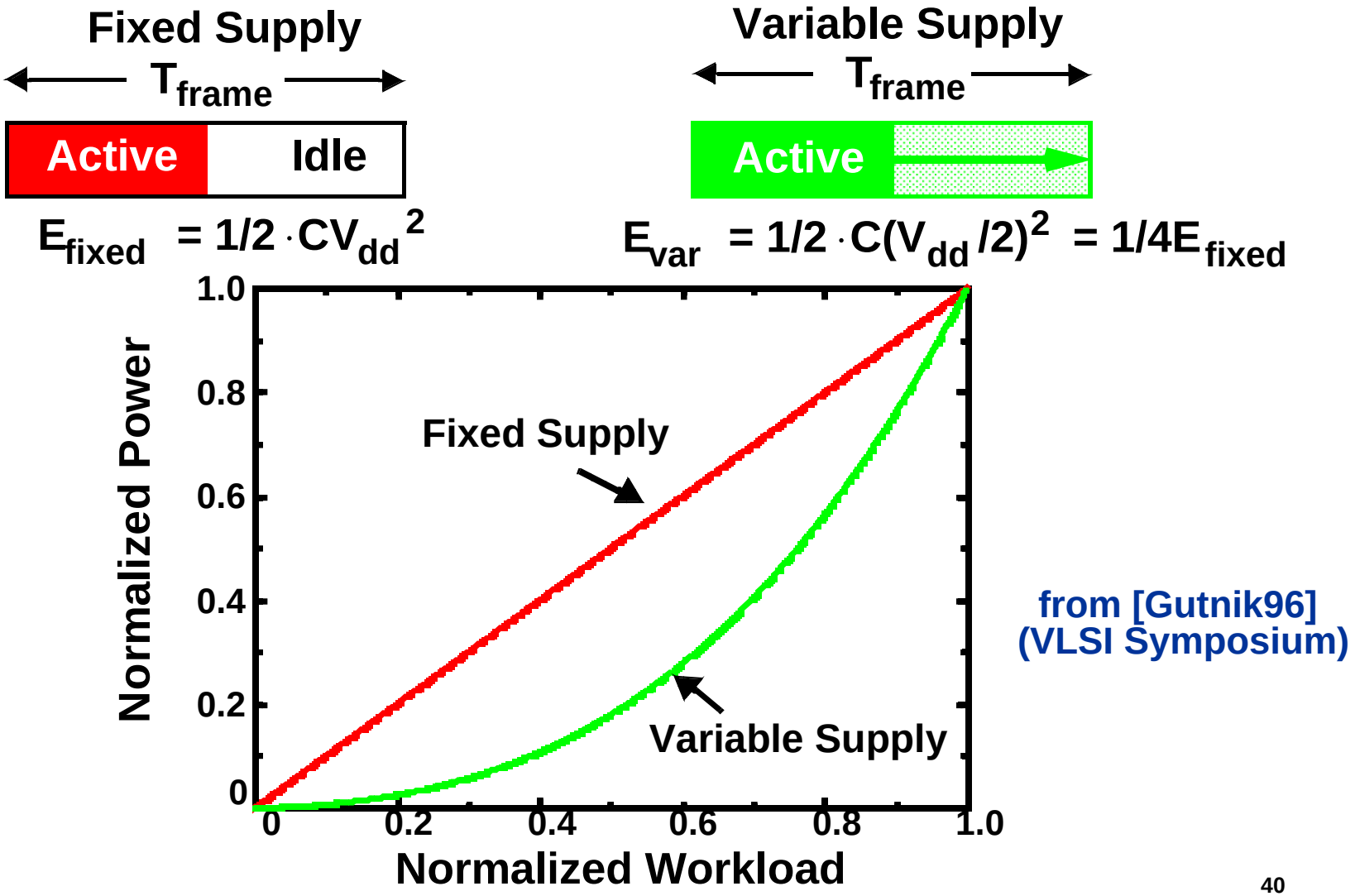
- **Issues:**
  - **Cost of restarting: latency vs. power trade-off**
    - **increase in latency (response time)**
    - **increase in power consumption due to startup**
  - **When to Shutdown:** *Optimal* vs. *Idle Time Threshold* vs. *Predictive*
  - **When to Wakeup:** *Optimal* vs. *On-demand* vs. *Predictive*
  - **Two main approaches: (Reactive versus Predictive)**
    - **"Go to Reduced Power Mode after the user has been idle for a few seconds/minutes, and restart on demand"**
    - **"Use computation history to predict whether $T_{block}[i]$ is large enough ($T_{block}[i] \geq T_{cost}$)"**

To Shutdown or Reduce Voltage?

- **Observation:**
  - **better to lower voltage than to shutdown in case of digital logic**
- **Example: task with 100ms deadline, requires 50ms CPU time at full speed**
  - **normal system gives 50ms computation, 50ms idle/stopped time**
  - **half speed/voltage system gives 100ms computation, 0ms idle**
  - **same number of CPU cycles but 1/4 energy reduction**

- **Voltage gets dictated by the tightest (critical) timing constraint both on throughput and latency --> dynamically change voltage**
  - **Use voltage to control the operating point on the power vs. speed curve**
    - **I.e., power and clock frequency are functions of voltage**
  - **Main challenge here is algorithmic:**
    - **one has to schedule the voltage variation as well!**
      - ➤**via compiler or OS or hardware**

Dynamic Voltage Scaling (DVS)

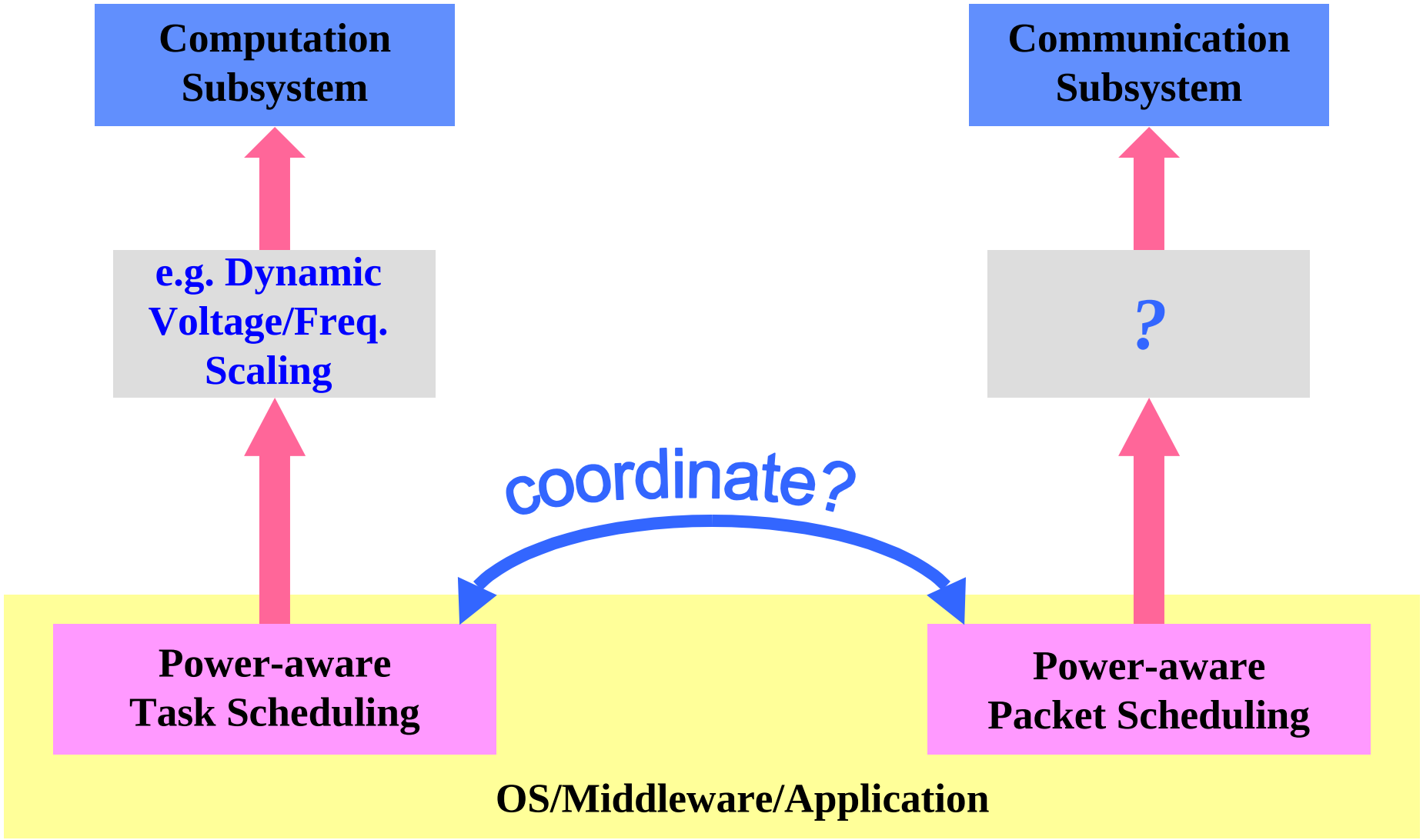


from [Gutnik96]
(VLSI Symposium)

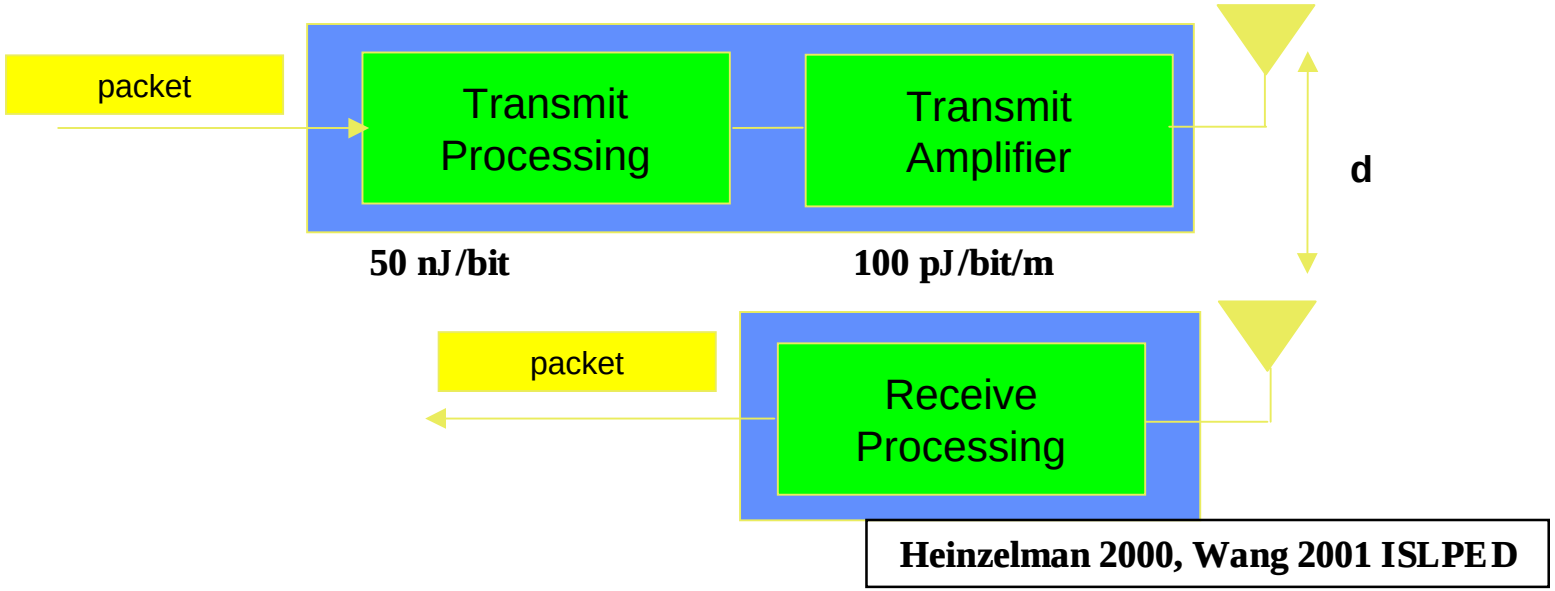
Wireless NES

- Components of power awareness in firmware design
  - signalling protocols, choice of modulation
    - e.g., On-Off Keying, ASK need only a threshold detector, whereas FSK needs a frequency discriminator (but less susceptible to noise)
  - transceiver architecture for high performance in E_b/N_0
  - RF, IF analog circuits
  - Baseband DSP
    - e.g., max. a posteriori (MAP) estimation, iterative (turbo) channel estimation and coding can reduce TX power by 1/3 to 1/2
- Increased signal processing in BB enables flexibility in RF/IF design
  - e.g., increased noise figure or reduced TX power
- Higher level protocols (e.g., multiple access, link layer) can enable power hungry circuits to be active for as short time as possible
  - e.g., MAC often includs some variant of TDMA
  - exploit asymmetry between basestation and mobile units (e.g., different duty cycles in paging receivers)

Power Management in Communication Subsystems



Radio Power Consumption



- The RF power dominates over the electronic power associated with transmit or receive processing, except for short distances (5-10 m).
- Voltage scaling and other low power logic techniques do not affect RF power consumption
- Need effective transmit power control by adapting to link conditions
- TX radios can be “duty cycled”, wakeup for RX radio may be an issue.

Digital Modulation & Demodulation

- **Modulation**
  - **maps sequence of “digital symbols” (groups of n bits) to sequence of “analog symbols” (signal waveforms of length TS)**
- **Demodulation**
  - **maps sequence of “corrupted analog symbols” to sequence “digital symbols” - e.g. maximum likelihood decision**
- **Coherent or Synchronous Detection**
  - **process received signal with a local carrier of the same frequency and phase**
  - **e.g. phase shift keying, frequency shift keying, amplitude shift keying, continuous phase modulation**
- **Noncoherent or Envelope Detection**
  - **requires no reference wave**
  - **e.g. FSK, differential PSK, CPM, ASK**

Selecting a Modulation Scheme

- Provides low bit error rates (BER) at low signal-to-noise ratios (SNR)
- Occupies minimal bandwidth
- Performs well in multipath fading
- Performs well in time varying channels (symbol timing jitter)
- Low carrier-to-cochannel interference ratio
- Low out of band radiation
- Low cost and easy to implement
- Constant or near-constant “envelope”
  - constant: only phase is modulated
    - may use efficient non-linear amplifiers
  - non-constant: phase and amplitude modulated
    - may need inefficient linear amplifiers

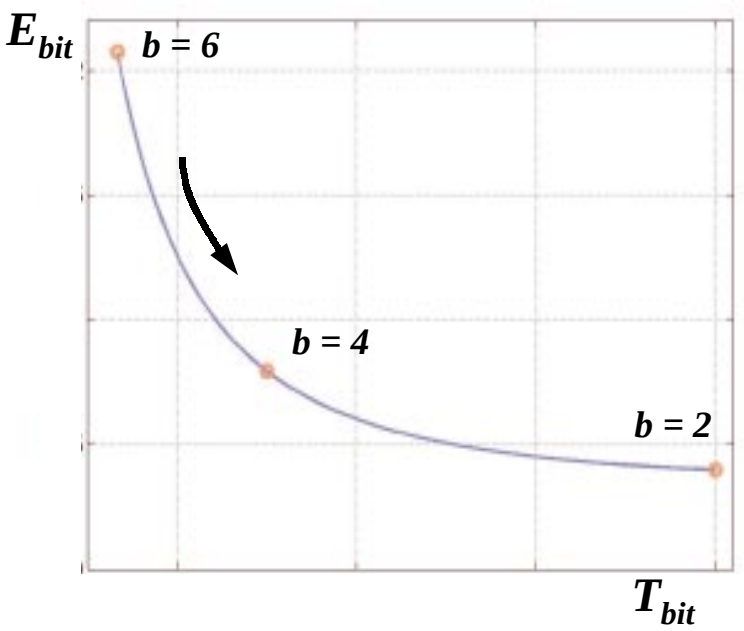
Choice of a modulation based on : energy efficiency Eb/N0 for a certain BER and bandwidth efficiency R_b/B

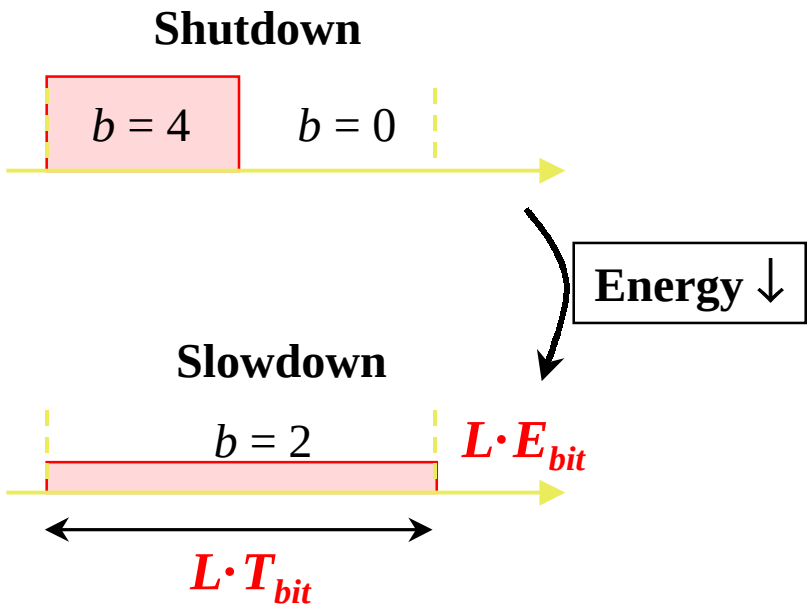
Dynamic Power Management in Radios

[Srivastava, UCLA]

- Two possible "knobs" to control power/performance tradeoffs:
  - Dynamic Modulation Scaling (DMS)
    - modulation is process of converting bits into channel symbols
    - modulation level b = # bits in a symbol
    - changing module level changes time and energy to send a bit for a given channel condition
  - Dynamic Code Scaling (DCS)
    - utilizes redundancy coding for combating error
    - change the code rate = size original data / size of coded data
    - change in code rate provides classical delay, energy tradeoff.
- These can be combined and should also be integrated with higher layer DPM policies (e.g., for packet scheduling)
  - packet scheduler in MAC decides when to send a packet
  - DMS/DCS can be based on number of packets in the queue

DMS

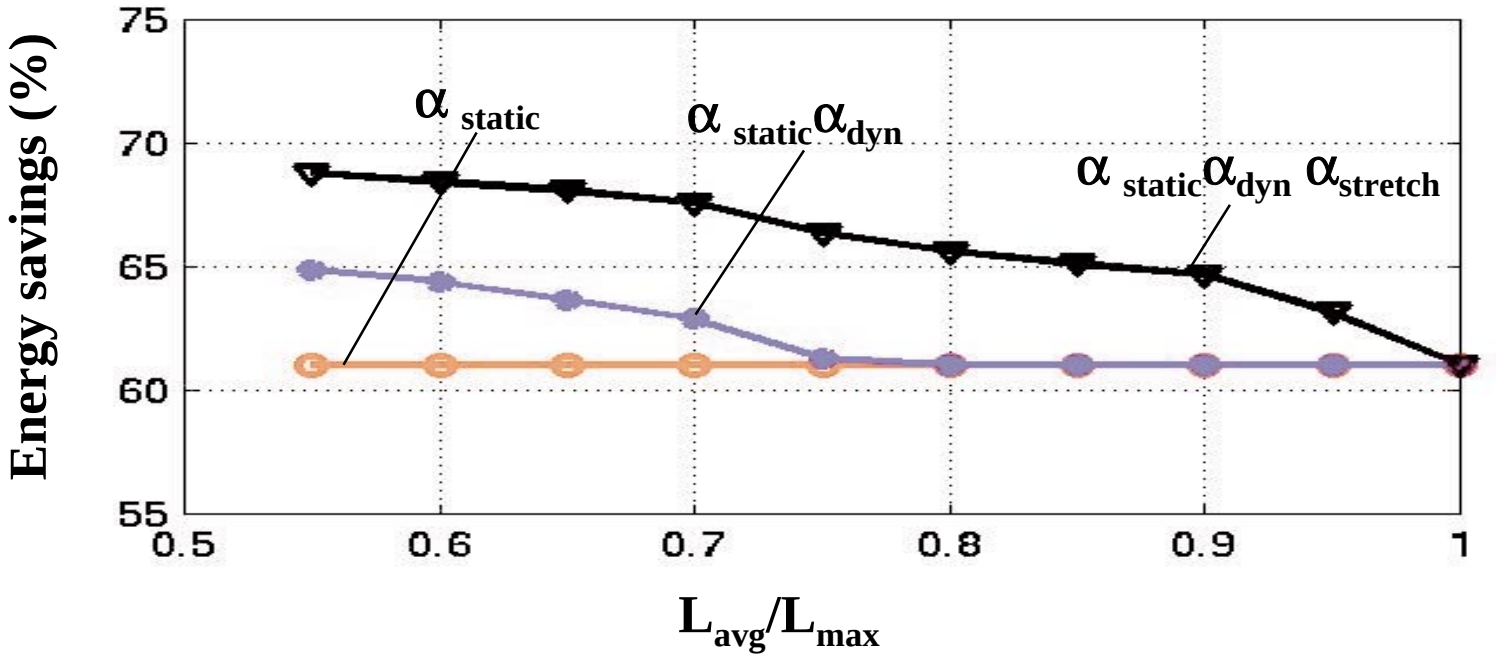




- **The energy - delay curve is** convex
  - ⇒ **Slowing down is more energy efficient than shutting down**
- **For energy efficiency, operate** as slow as possible

Energy-aware Real-time Packet Scheduling

- Analogous to RTOS task scheduling
- Exploit variation in packet length to perform aggressive DMS



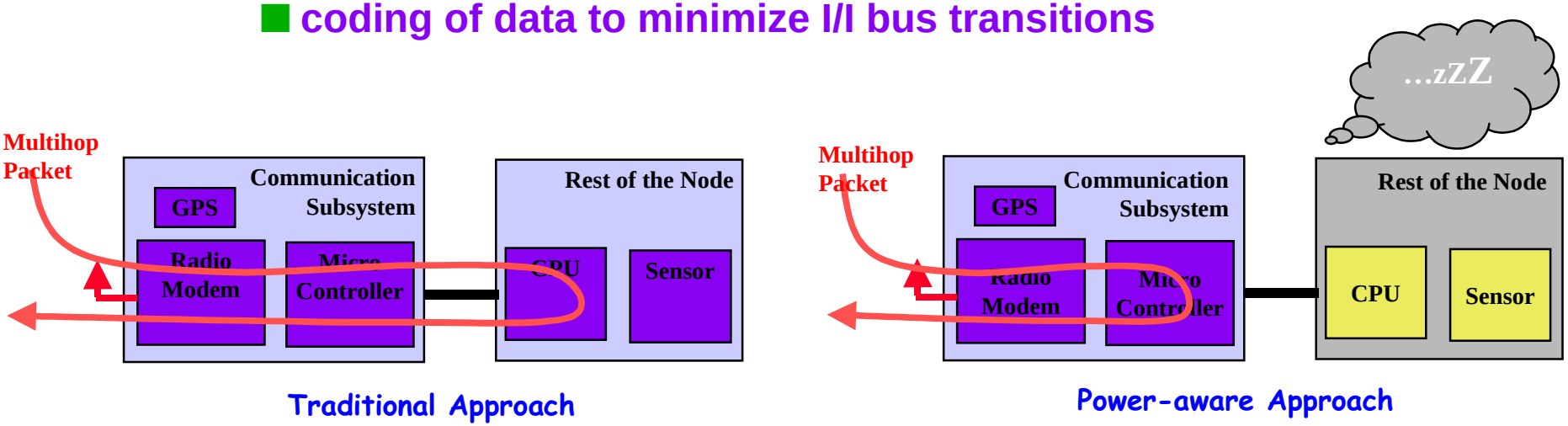
➨ Up to 69% reduction in transmission energy (Srivastava, UCLA)

Help from Upper Layers in Power Management of Wireless NES

- Minimizing idle time matters the most
  - other factors secondary, such as specific protocol
- Transport: don't leave the receiver idle while there is congestion in the network
- Data scheduling: coordinate data delivery to receiver in bursts
- S/W control of NI for application-level optimizations

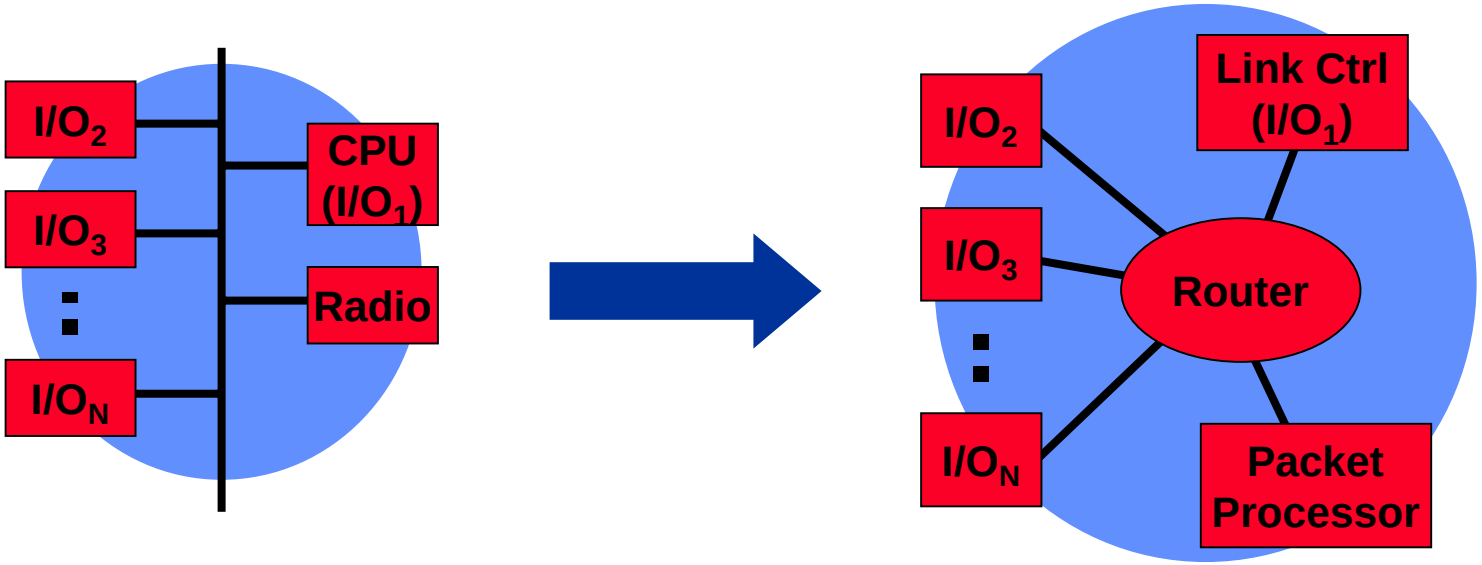
Architectural Strategies

- Gated-clocks and power-shutdown
  - stopping unused hardware
- More efficient algorithms and architectures
  - focus on power under a speed constraint
- Proper I/O interconnect design and packaging
  - include as much of system in a single package
  - recompute data rather than refetch from memory
  - use local memory / cache to minimize I/O
  - coding of data to minimize I/I bus transitions



Source: Mani Srivastava, UCLA

Energy Impact of Architecture: Shared-bus vs. Switched



- Router-based approach isolates I/O devices
  - reduces switching capacitance, frequency, voltage, and ultimately energy consumption
  - takes the main CPU out of the datapath
  - allows rapid, low power, LUT-based decision making up through the network layer

Energy Efficient Protocol Processing

- Circuit level low power circuits address PHY layer implementation
- The problem is that for networked systems, high level protocols often implicitly assume continuous availability of lower layer (especially PHY layer) functions
- Effective power management requires power aware protocol design
  - e.g., 802.11 MAC can reduce power by allowing both PHY transmitter and receiver to be turned off without a station appearing as disconnected from LAN
- In the following we address:
  - Energy efficiency in MAC and Link layers
  - Energy efficient higher layer processing

Energy Efficient MAC

- Reduce time radio is in transmit mode
  - minimize random access collisions and consequent retransmissions
  - use polling, slot reservation
- Reduce time radio is in receive mode
  - minimize listening for packets to arrive
  - broadcast periodic “schedule” telling receivers when to wake up
- Reduce transmit-receive and on-off turn-around
  - maximize contiguous transmission slots from a radio
- Allow mobiles to voluntarily enter into sleep mode
- Reduce MAC signaling traffic

The MAC Level Perspective: Optimized use of PHY Layer

- Higher power consumption at higher symbol rates
  - power hungry equalizer is needed to combat ISI
  - modulation requiring higher Eb/N0
    - e.g. higher order QAM
- Power hungry DSP to combat impairments
  - channel coding
  - RAKE receiver
  - antenna array
- Proper use of PHY layer processing is critical
  - adapt according to conditions

The MAC Level Perspective
Optimized MAC Protocol

- Simple vs. complex protocols
  - computation + communication energy per “useful” bit
  - asymmetric protocols to keep mobile simple
- Frame structure
  - header overhead: e.g. using header compression
  - encoding header for lower energy decoding, and invoking costlier receiver functions on the frame body only if the frame is for the receiver
    - e.g. in HIPERLAN header is at lower rate (no equalizer)
- Adapting frame length

The MAC Level Perspective
Optimized MAC Protocol (contd.)

- Radio state management (active vs. sleep)
  - how to send packets to a receiver that sleeps
  - how to make sure not to miss packets
  - impact on higher layer protocols
- MAC-level error control
  - adapting FEC according to channel conditions
  - channel-state dependent scheduling
  - transmission channel probing during ARQ
    - channel state may be persistent
    - probe impaired channels via short low-power probes instead of blind retransmission of high-power data packets

The MAC Level Perspective Optimized Network Design

- Cell size
  - reducing cell size means smaller transmit power
    - system capacity also goes up
  - but complexity of mobility management goes up
    - more hand-off events
- Centralized vs. ad hoc network architecture
  - networks with basestations
    - asymmetric processing with complexity kept at BS
    - but, make intra-cell communication less efficient
  - ad hoc networks
    - multi-hop takes less energy than single hop
    - intermediate nodes lose battery energy to other's traffic

Power Aware Routing & Channel Allocation

- Minimize transmit power of mobile nodes to increase lifetime of individual nodes and network
  - minimize total power of n/w, or max power by a node
  - Note: Least-power path != shortest path
- Conventional multihop routing protocols such as DSR, DSDV etc. are power-unaware
    - metrics used: shortest hop, shortest delay, link quality, location stability, message & time overhead
- Metrics to consider
  - Minimize energy consumed / packet
    - ➤large dissipation at selected bottleneck nodes
  - Maximize time to network partition
    - ➤important for sensor networks etc.
    - ➤load balance across the nodes in the cut-set
    - ➤difficult to implement
  - Minimize variance in node power levels
    - ➤no single node is penalized
    - ➤difficult to implement

Energy Efficiency at The Transport Layer

- Reported 48-83% power savings in wireless NIC power with additional delays of .4-3.1s
- Idea #1: data and header reduction
  - use compression to reduce communication time
  - communication vs. computation power
- Idea #2: shutdown
  - selectively choose short periods of time to suspend communications and shut down the wireless NIC
  - queue data for future delivery during shutdown
  - decide when to restart
  - trade-off between power consumption and delay

Why do Shutdown at The Transport Layer?

- Lots of challenges
  - a node can only guess when data is destined for it
    - no knowledge as in MAC-layer schemes
  - side effects on other hosts
    - sender may have buffer overflows
    - sender may waste power trying to communicate with sleeping node
  - balancing power savings against delay, and additional consumption in shutdown and restart
    - when to shutdown and when to restart
- But, one crucial advantage:
  - opportunity to incorporate application knowledge to balance power savings and data delay in an end-to-end fashion.

Energy Efficient I//O Encoding

- C of system busses is >> C inside chips
  - large amount of power goes to I/O interfaces
    - 10-15% in uPs, 25-50% in FPGAs, 50-80% in logic
  - encoding bus data can reduce the power significantly
    - but need to handle encoding/decoding cost (power, latency)
- Examples:
  - Gray code and T0 code on address busses
    - addresses usually increment sequentially by 1
  - Compression to remove redundancy
  - Bus-Invert Coding
    - transmit D or invert(W), whichever results in fewer transitions from the previous transmitted code
    - an extra signal indicates polarity
    - works better for small N (25% for N=2, 18.2% for N=8, 14.6% for N=16) … use k sub-busses with k polarity bits!

Example: UCLA/RSC Protocol Suite

- For use in *ad hoc* sensor networks.
- A TDMA-like schedule for nodes with no neighbor awareness initially
- Synchronization among neighbors and channel assignment done together
  - Ad hoc network acquisition
    - "superframes" contain a number of invitation frames and the rest are for synchronized communication
    - connections built from node-to-node, node-to-network, network-to-network sequentially such that the fraction of invitation frame decreases
  - Routing protocol
    - build overlapping spanning trees outward from gateways -- constructed by messages that keep track of hops they have moved through
    - nodes are hierarchically arranged: tree built from connections from one tier to only a lower level.

Network Level Power Control

- Dynamic power control in cellular networks pursued diligently for several reasons beyond node battery life
  - ensure a received signal power level that accounts for changes in channel conditions due to node mobility
    - attenuation, reflection, diffraction, scattering, multipath
  - minimize TX power to reduce cochannel interference, I.e., interference with channels on the same frequency in remote cells
  - for CDMA, equalize power received from all mobile units at the basestation (BS) for good system performance
- Power control can be based on
  - SIR, BER or Signal Strength
    - Interference form channel noise, multi-user
      - ➤sometimes can cause positive feedback
- Power control can be
  - centralized or distributed
  - open-loop or closed-loop

Open-Loop Power Control

- Also known as autonomous power control
- Depends solely on the mobile unit with no feedback to the BS
- The BS continuously transmits a pilot (unmodulated signal)
  - pilot allows a mobile unit to acquire the timing of the forward (BS to mobile) channel and provides a phase reference for demodulation
- The mobile unit monitors the received power level of the pilot and sets the transmitted power in the reverse (mobile to BS) inversely proportional to it
  - assumes correlated forward and reverse channel conditions
- Open loop is not as accurate but it enables quick reaction to rapid fluctuations in channel conditions
  - especially important in CDMA since sudden changes in received signal strength at BS may drown out all other signals at the same frequencies (combat near-far effect)

Closed-Loop Power Control

- Here, the mobile unit adjusts the transmitted signal strength (to the BS) based on conditions in *that* reverse channel such as received signal power level, received SNR, or received BER
- The BS makes power adjustment decision and communicates a power adjustment command to the mobile unit on a control channel
- Closed-loop can also be used for power adjustment in the forward channel (BS to Mobile)
  - in this case the mobile provides information about received signal quality to the BS which then adjusts the transmitted power
- Example: GSM

Example: GSM TX Classes

Power Class	BS Power (W)	Mobile unit (W)
1	320	20
2	160	8
3	80	5
4	40	2
5	20	0.8
6	10	
7	5	
8	2.5	

Power Control in WCDMA

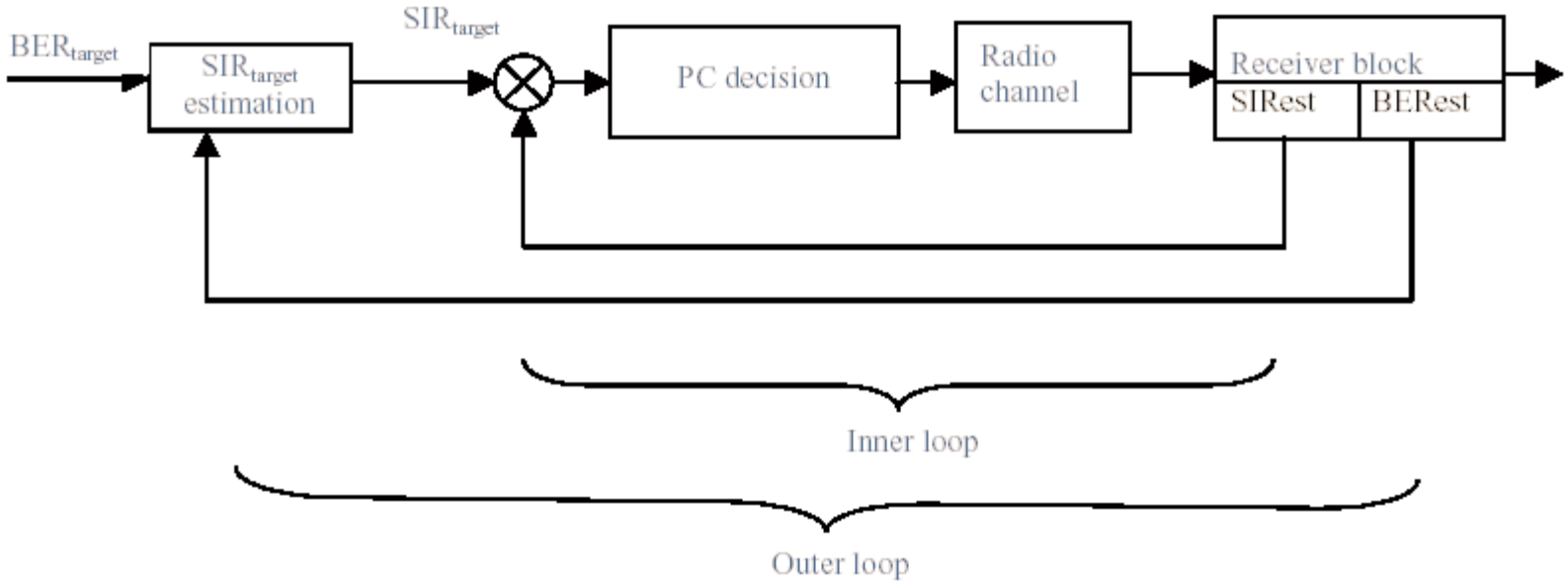
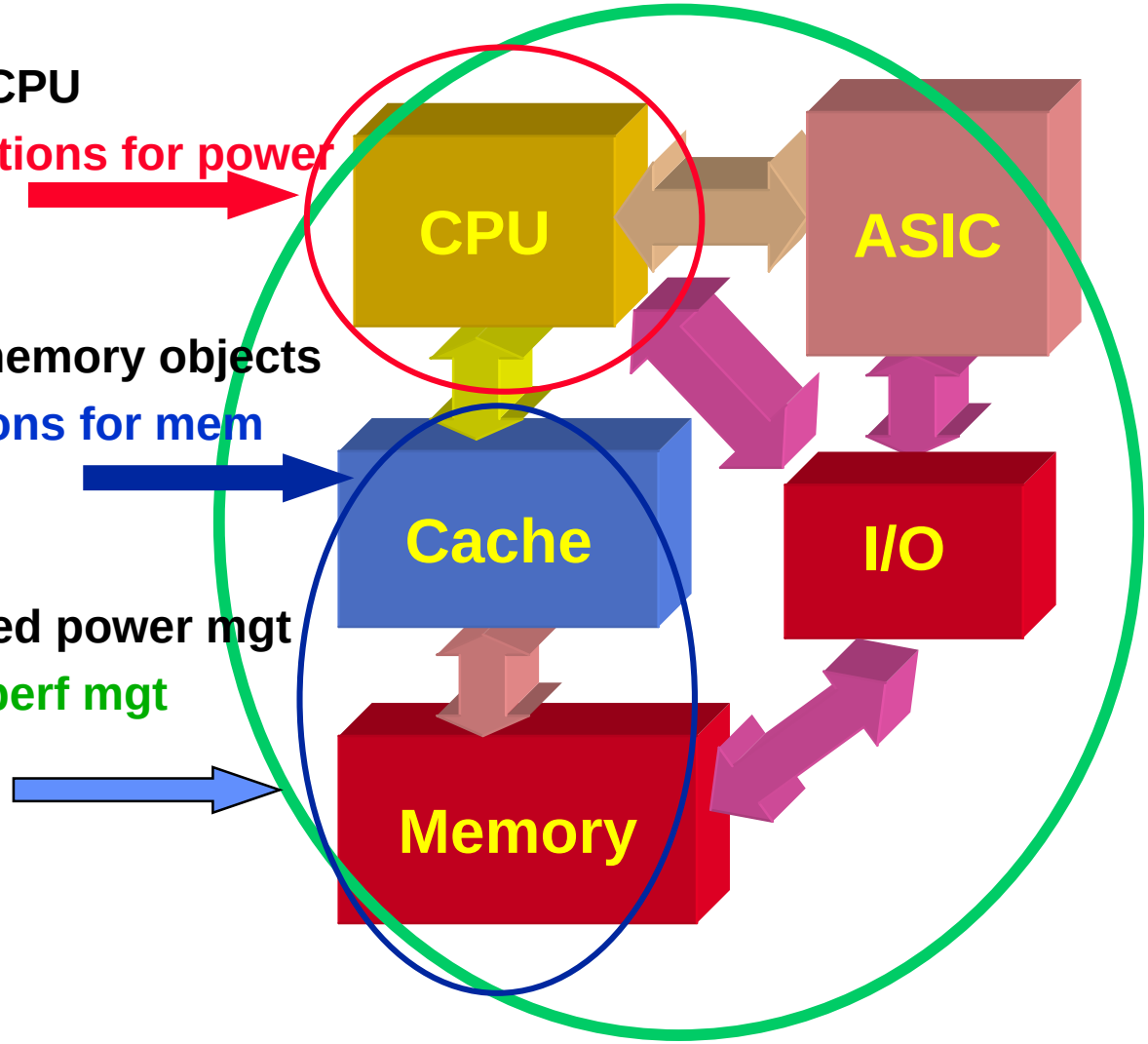


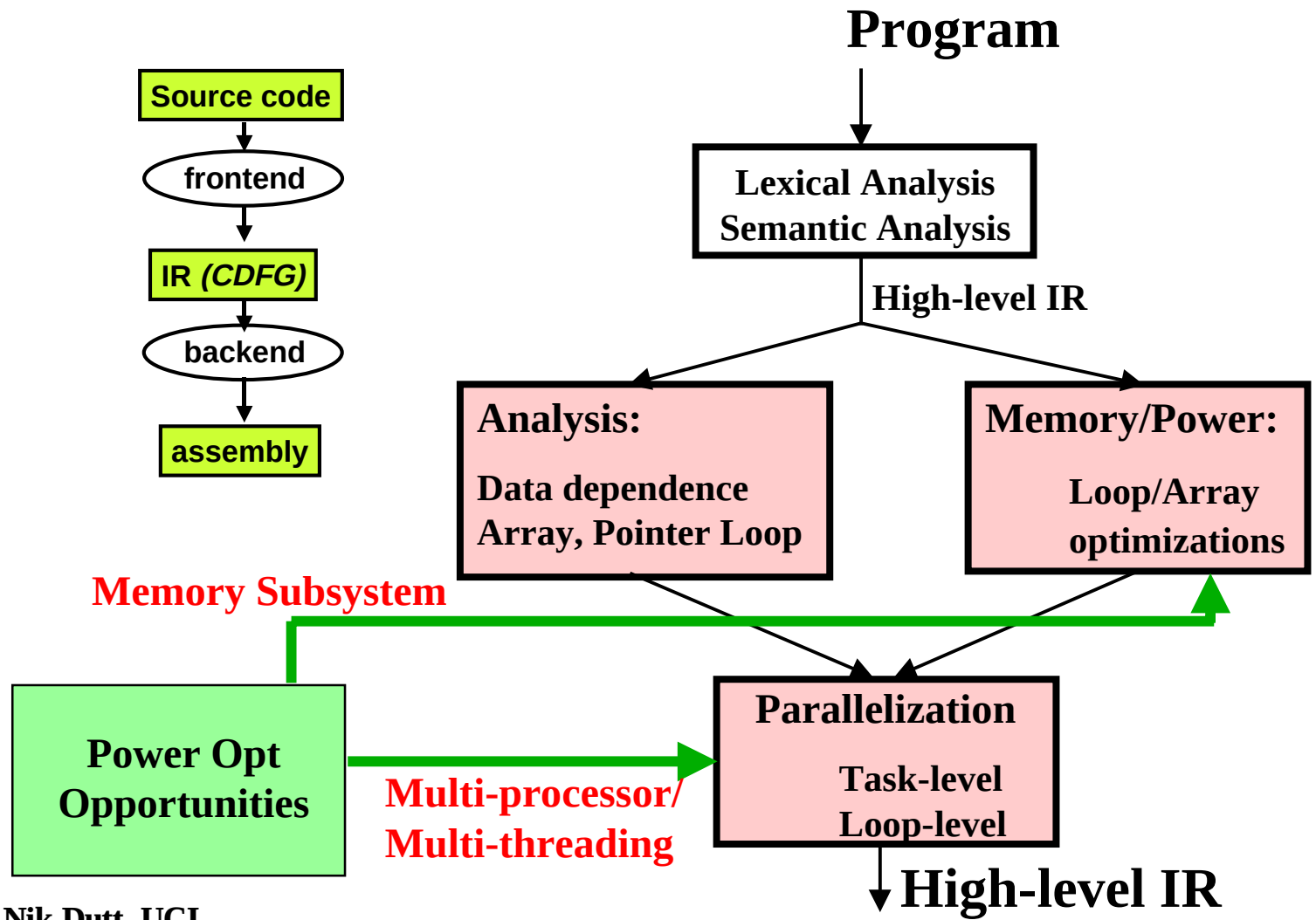
Fig. 1. Power control in WCDMA system. In the receiver block, the received SIR and BER are estimated and used respectively for the inner-loop and the outer-loop.

Software Power Optimizations

- Code running on CPU
  - Code optimizations for power
- Code accessing memory objects
  - SW optimizations for mem
- Compiler-supported power mgt
  - Dynamic pwr/perf mgt

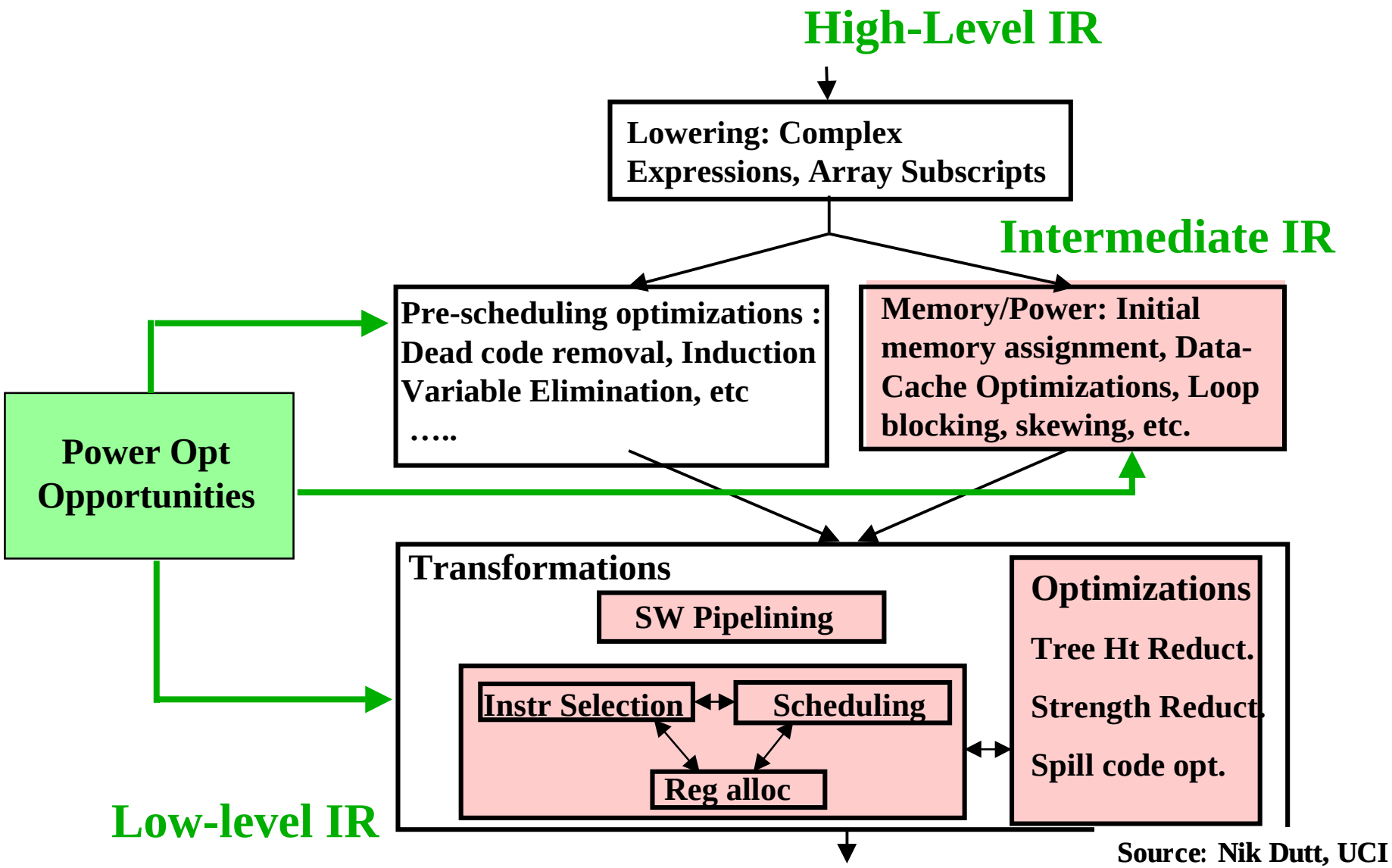


Compiler Flow (Front-End)



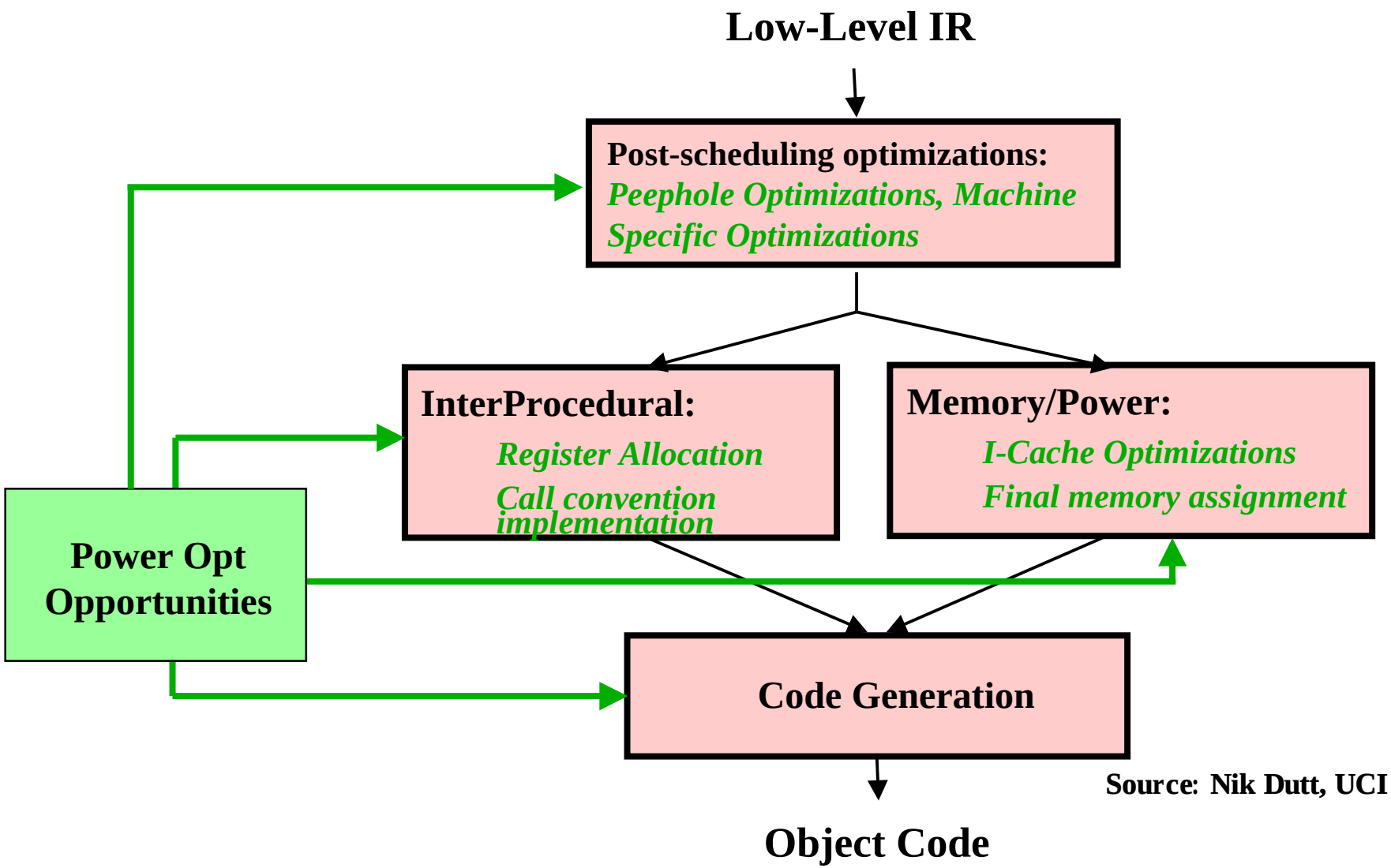
Source: Nik Dutt, UCI

Compiler Flow (Middle)



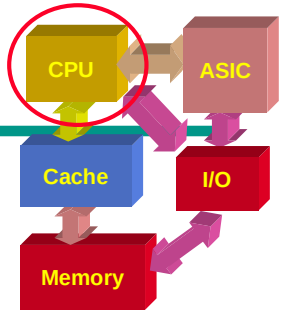
Source: Nik Dutt, UCI

Compiler Flow (Back-End)

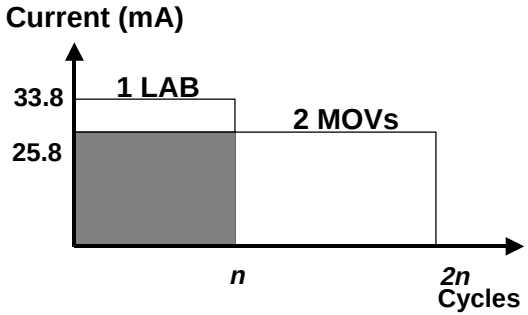


Source: Nik Dutt, UCI

Energy-efficient Software Generation

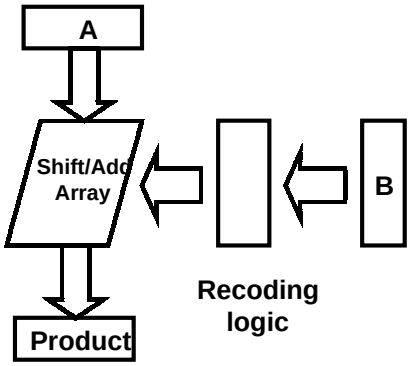


- Compiling for Speed is Good
  - Faster Program => Lesser Energy (with Power mgmt)



- Dual memory loads; Instruction packing (DSP)
  - Two on-chip memory banks
    - Dual load vs. two single loads
  - 1-cycle Packed vs. 2-cycle unpacked
  - Almost 50% reduction in energy

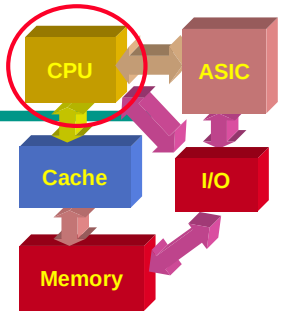
- Reorder Instructions to reduce switching effects
  - Not much impact on large general purpose CPUs
  - Useful in DSPs - (~15% benefit) [Lee et. al. TVLSI, Dec '96]



- Swapping multiplication operands (DSP)
  - Put operand with lower weight in B (upto 30% red)

[Source: Tiwari]

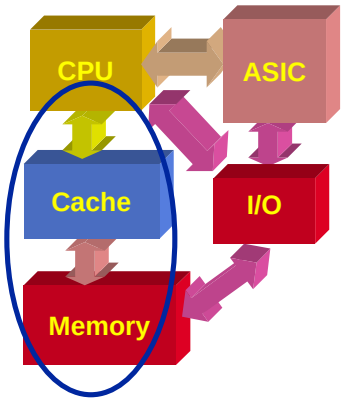
Instruction Scheduling and Reordering



- Power depends on switching activity, units accessed
- Power-driven scheduling
  - scheduling to reduce pipeline stalls
  - selecting a minimum-power instruction mix for executing an application
  - reducing switching on address/data lines
    - instruction reordering
      - ➤pairs of instructions have different power consumptions
    - operand swapping
  - low-power instruction sets
  - instruction packing
  - shut down unused units

Register Optimizations

- Register files size is increasing for newer processors
  - power consumption of registers – important component of the overall power
  - still power per access less than for cache or memory
    - tradeoffs between register file and cache/memory
  - special techniques for optimizing register file power consumption
- More aggressive register allocation techniques
  - better utilization of registers
- Techniques for minimizing switching activity



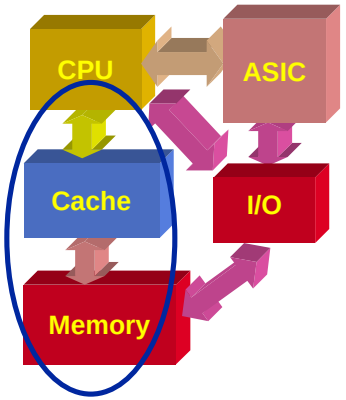
Register Optimizations

[Tiwari J. VLSI Signal Proc. Aug '96] Software Energy Optimization

- Reduce Memory Accesses, Make better use of Registers
  - Data for i486
  - register access = 300 mA/cycle
  - memory read (cache hit) = 430 mA/cycle
  - memory write (write-through cache) = 530 mA/cycle

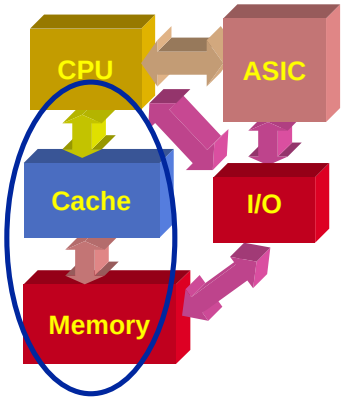
- can be achieved for e.g. by saving the least amount of context during function calls (compiler policies)

- better utilization of registers
  - optimal register allocation of temporaries
  - global register allocation for the most used variables
    - ➤use register operands as opposed to memory operands



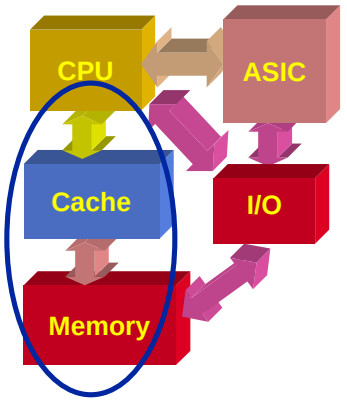
Memory Hierarchy Optimizations

- Memory – generally the most power hungry subsystem
- Study power consumption across the entire hierarchy
- Optimizations for cache
  - Code transformation to improve cache hit rates
    - reordering of memory accesses
    - allocation, blocking and copying of data
  - Partition memory into cached and direct-mapped (scratch-pad) spaces



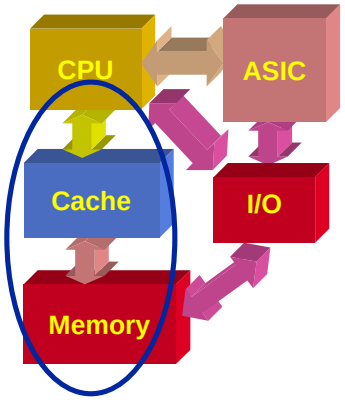
Memory Hierarchy Optimizations (Cont'd)

- Exploit data locality
  - reducing memory accesses by coalescing narrow memory references into wide ones
  - data transfer and placement
- Memory subsystem
  - memory bank assignment for low energy
- Reduce memory accesses
  - better utilization of registers
    - more aggressive register allocation
    - allocate local and global variables to registers



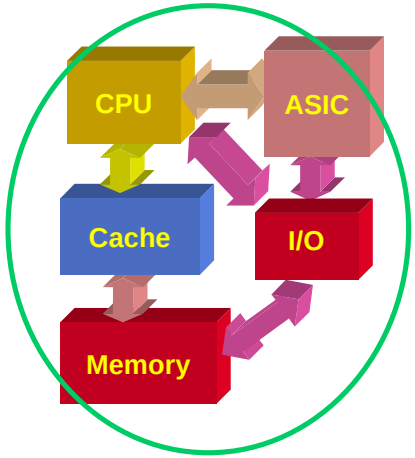
Memory Hierarchy Optimizations

- **Influence of Compiler Optimizations on System Power [Kandemir, DAC2000]**
  - linear loop transformations
    - loop interchange: improve data locality
    - some loop transformations may increase the power consumed in the datapath
  - loop tiling (blocking)
    - decrease power in memory, but may increase it in the core
  - loop unrolling
    - fewer memory accesses
    - reduction in power consumed in register file and data buses
  - loop fusion
    - reduce memory power consumption
  - loop fision
    - can increase memory power consumption
  - scalar expansion
    - power increase in core and memory system



Compiler-Controller Power Mgt.

- Understanding power profile is key
  - focus on major sources of power consumption
  - manage power
- Possible approaches
  - reduce average power consumption
  - modulate power consumption
    - over time
    - over resources
  - Examples: Transmeta, UCI COPPER



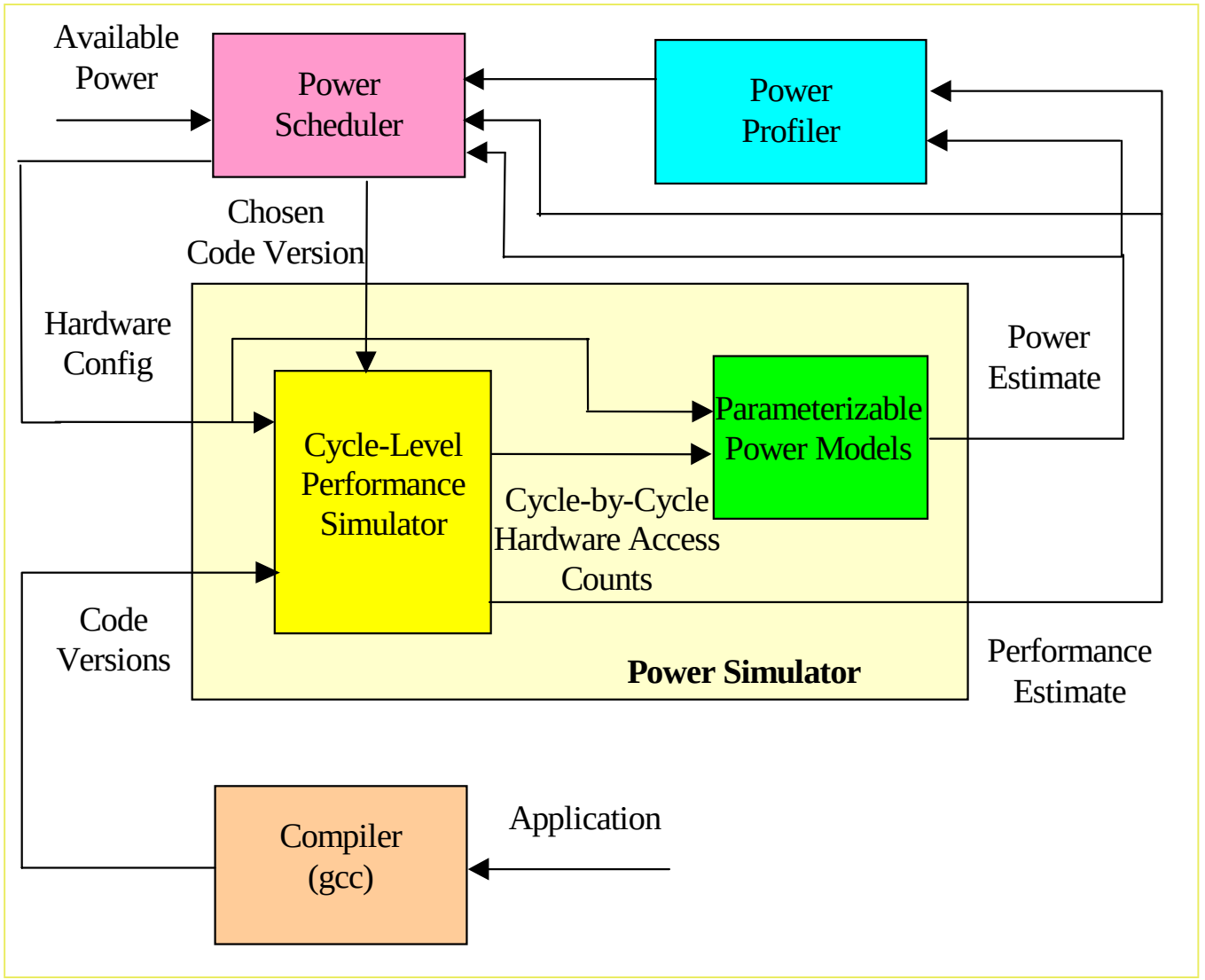
The COPPER Project

- Compiler-controlled Power-Performance Management
- Develop efficient architectural support and compiler techniques for power management
  - continuously -- as an application runs
  - targeted for high performance/VLIW machines
- Coordinated management of multiple techniques
  - reduction in power with little or no loss of performance.
- Develop techniques for dynamic compilation to actively trade off performance and power consumption
- Develop a retargetable, ADL-based, power-aware system simulation capability.

Approach

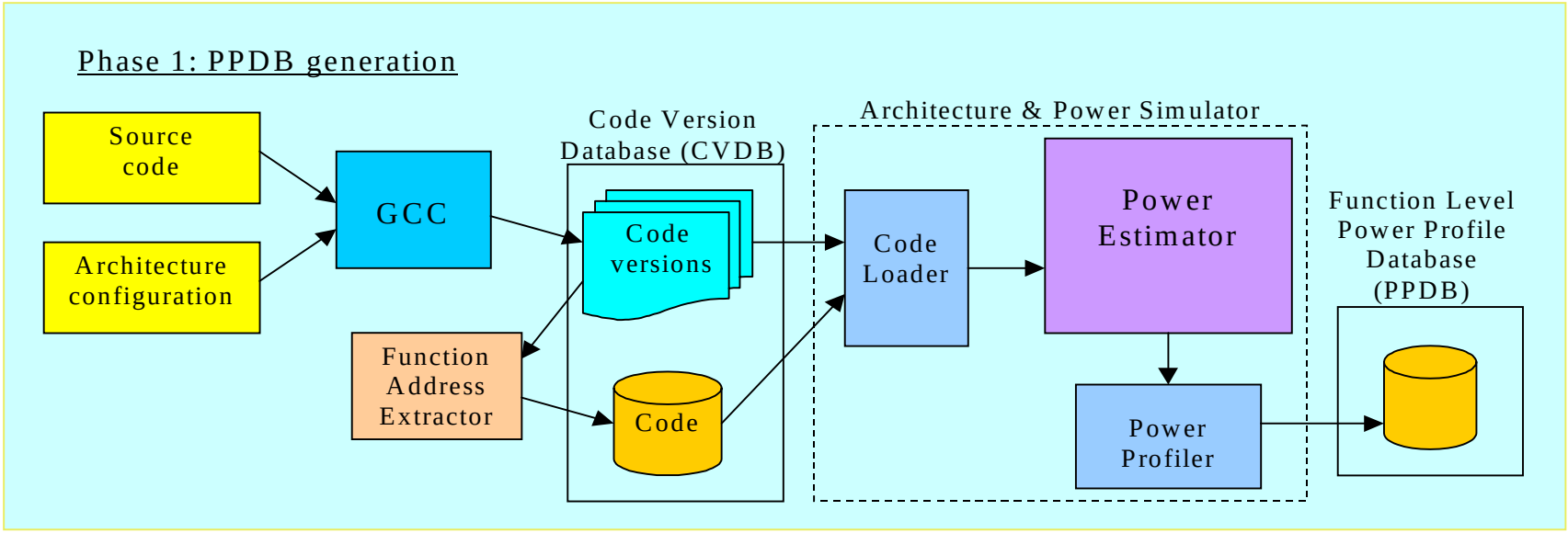
- Compiler Strategies for Power Management
  - Compiler-directed architectural "configuration"
    - generate "configuration code" embedded in the application
    - code "adapts" to new architectural organization at runtime
      - ➤JIT vs multi-version compilation techniques
      - ➤dynamic, on-demand optimization
  - Code annotation for dynamic compilation
    - trade-off compilation overhead for quality of generated code
  - Power-use Estimation for Compiler Control
    - static analysis to select "optimal" configuration
    - profile-based selection techniques
    - static or dynamic prediction methods

COPPER Framework

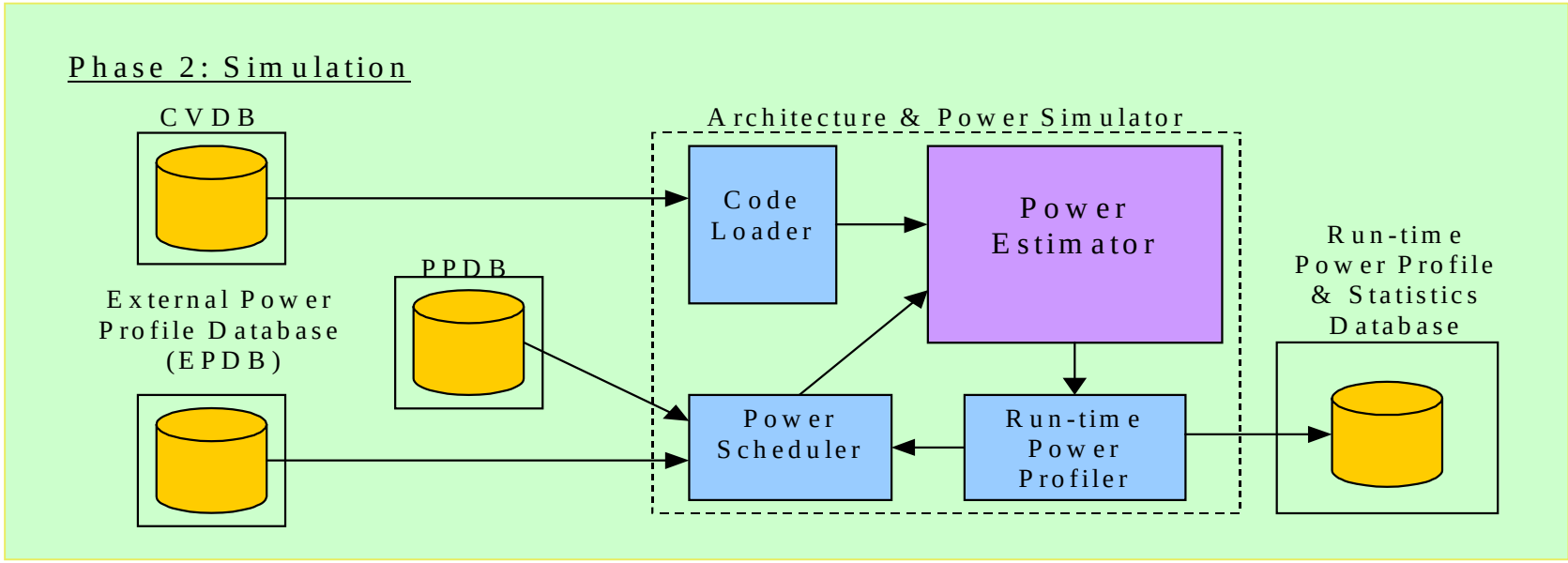


Step 1: Profile functions for energy use

- **Using Code Versions and Annotations**



Step 2: Use the profiles in the scheduling process



Baseline Architecture

- A MIPS R10K like processor
  - 4-wide issue, out-of-order (OOO) processor
    - 5-stage pipeline: fetch, dispatch, issue, writeback, commit
  - 32b integers, 64b f.p. numbers
  - register files: 32 integer and 32 FP registers
  - 32K L1 instruction cache, 32K L1 data cache
    - 32B L1 line size,
  - 512K L2 unified cache
    - 64B L2 line size
  - 2 int ALUs, 1 FP adder, 1 FP multiplier
  - 512-entry BTB, 2K entry branch predictor

Power/Performance "Knobs" Explored

❶ Memory hierarchy

❷ Instruction issue logic & issue width for VLIW m/c

❸ Dynamic Register File Reconfiguration

❹ Frequency and Voltage scaling

Dynamic Register Reconfiguration

- Compiler generates different *code versions*
  - Code versions have different ILP, register need…
- Power-performance profiling compiler decides on best code version

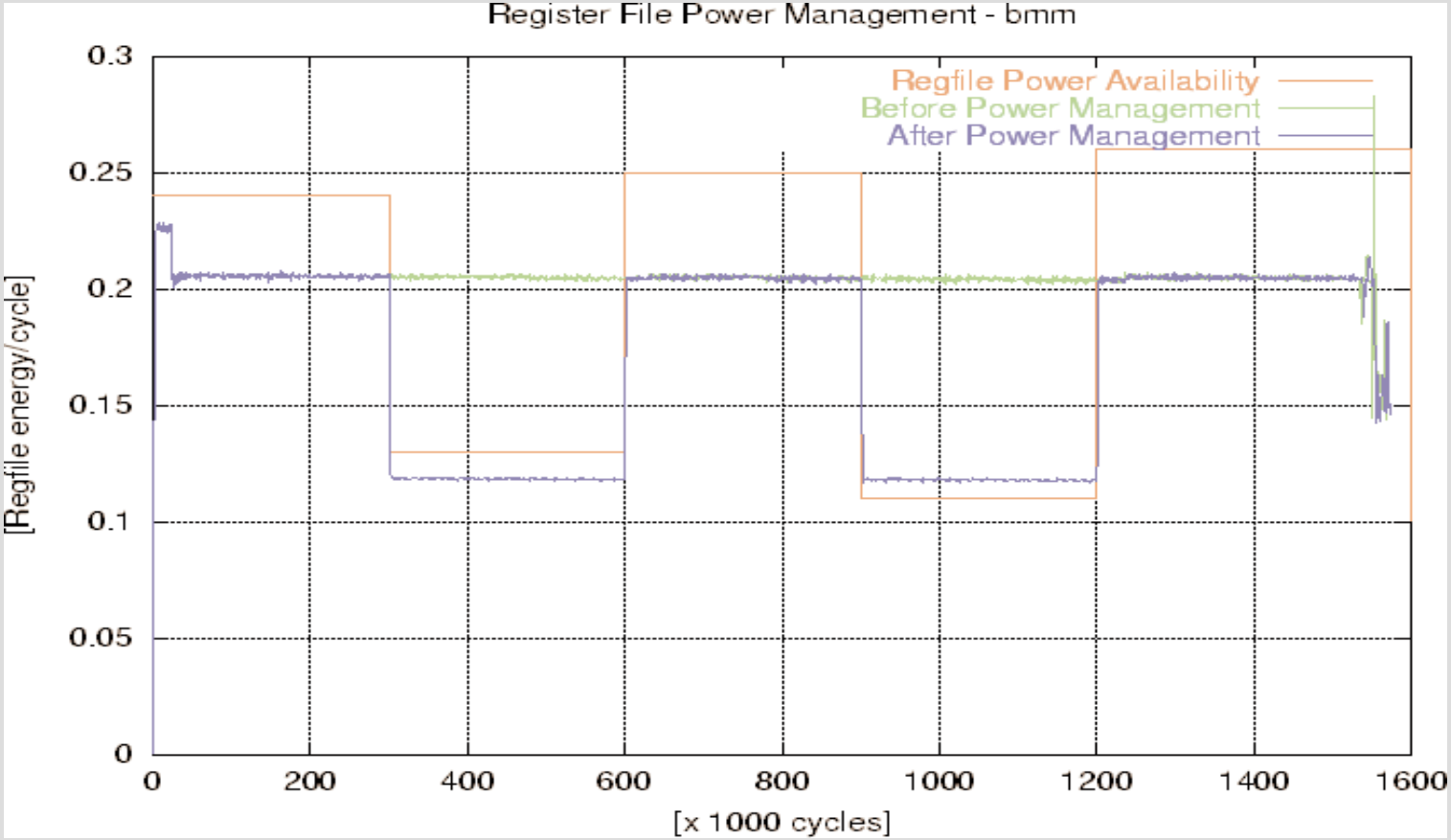
- Compiler generates *function code annotation*
  - carrying the chosen code version
  - carrying the number of registers needed for each code version
- At function calls, the run-time scheduler selects code version and adjusts register file size accordingly
  - all based on code annotation information

Power Scheduling Heuristic

- Select code version dissipating below and closest to the limit
- Switch to selected version and re-configure registers file
- Invoke every N cycles to continuously track energy use

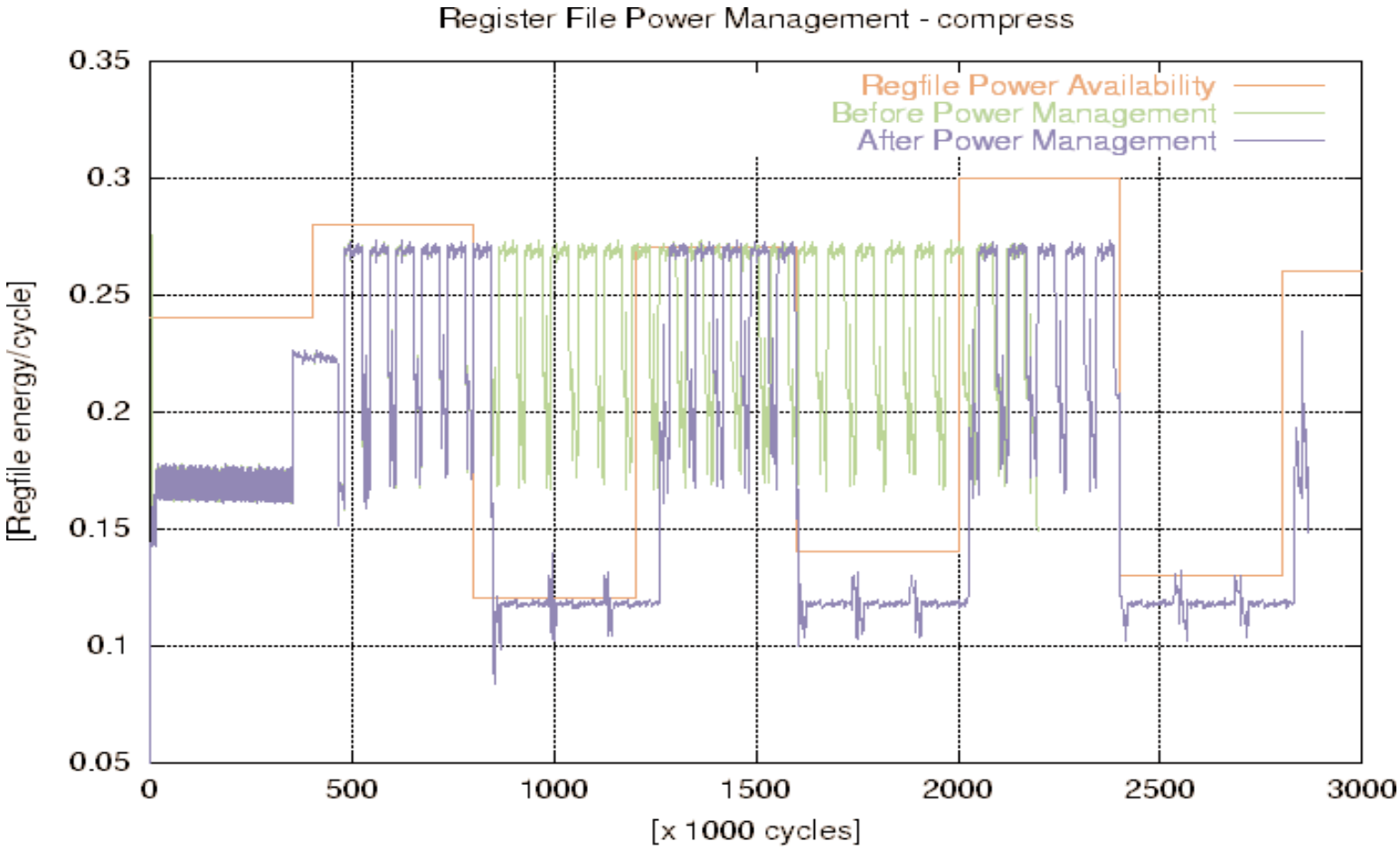
Power Management Through DRR

Register file power management without performance degradation



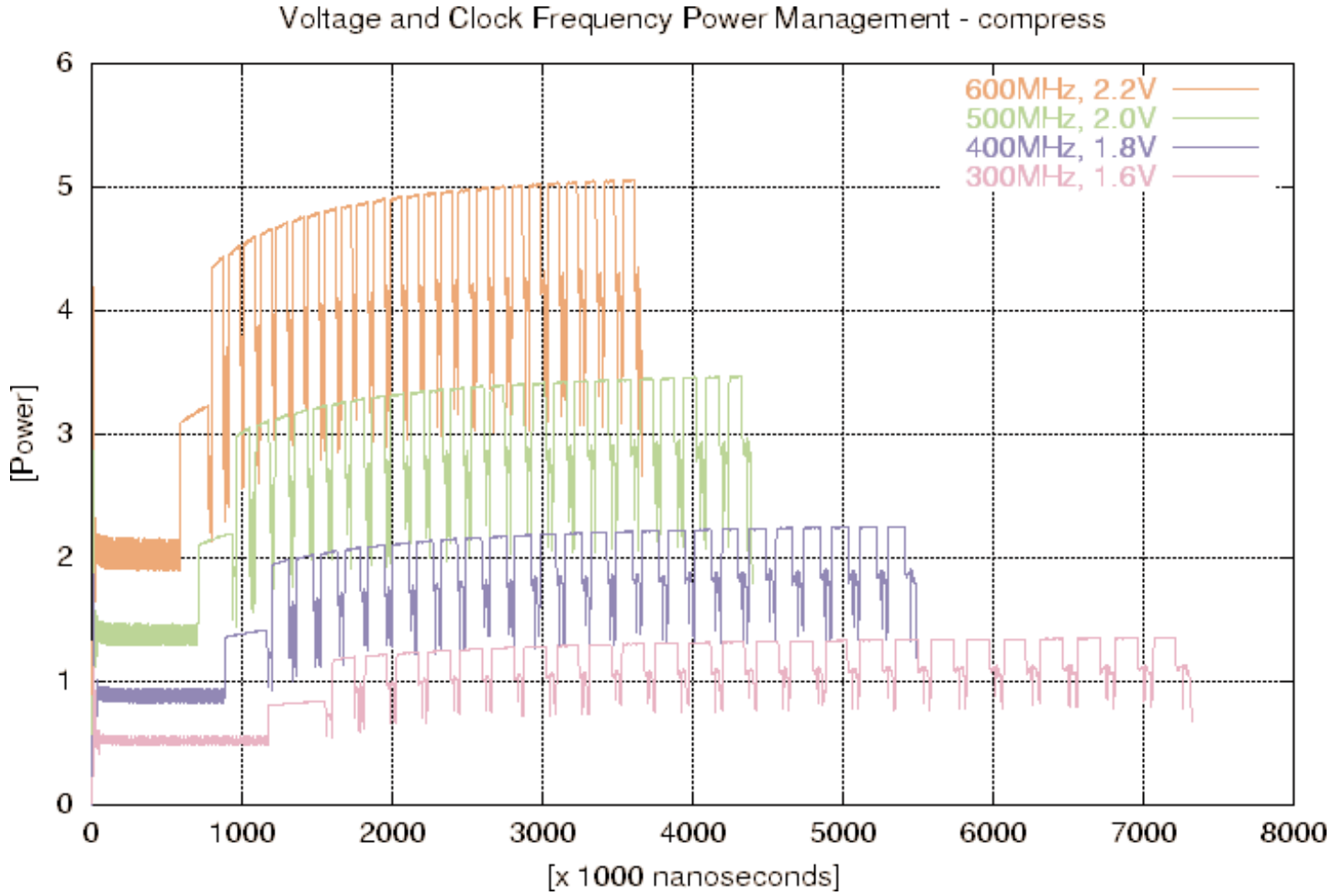
Unconstrained DRR

- **Register file power management with performance degradation**

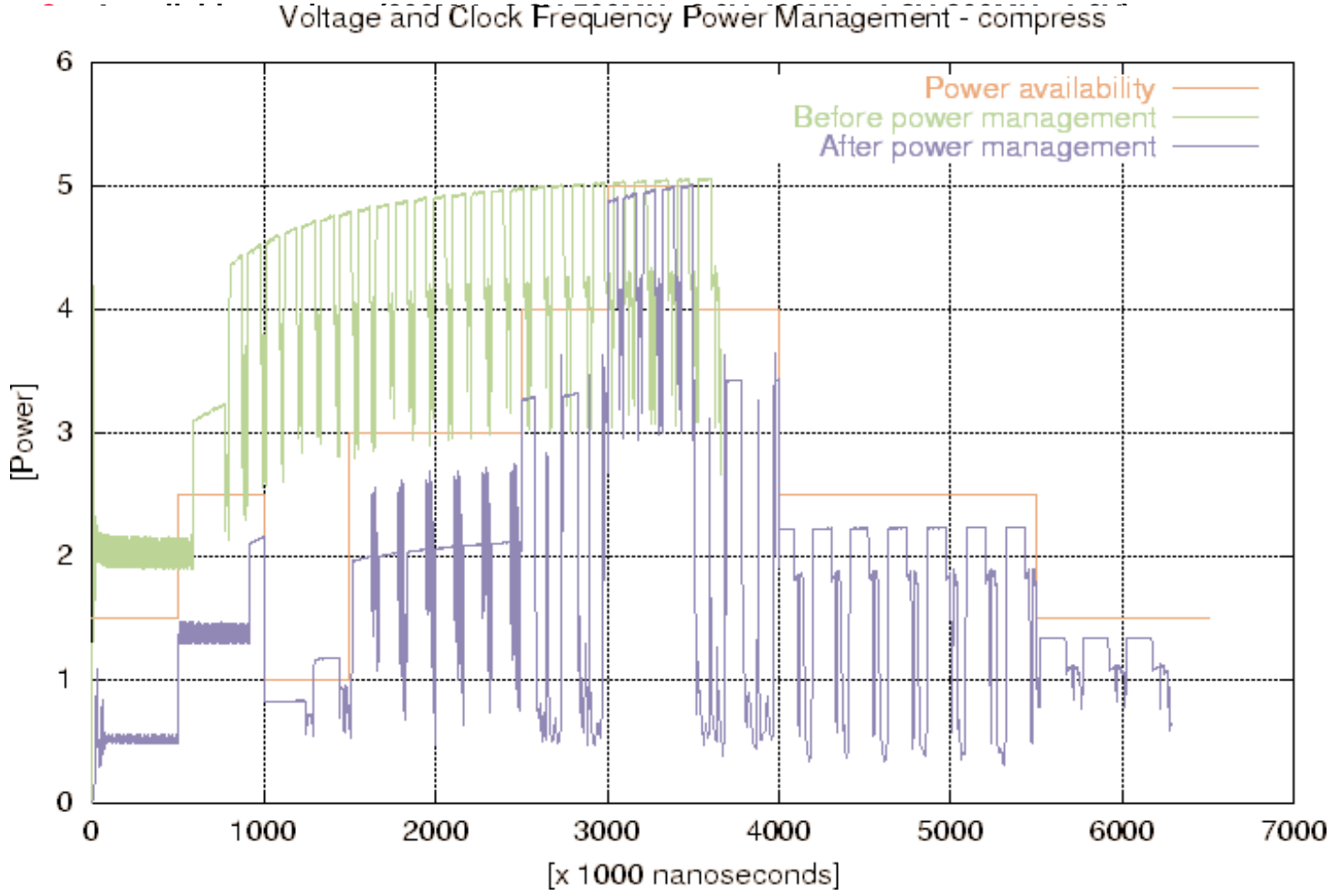


Frequency and Voltage Scaling

- **Code profiled for 4 clock frequency/voltage scaling configurations**



Power Management by F/V Scaling



Timing Constraints

- Consider timing constraints as bounds on operation intervals
  - upper and lower bounds
  - (determination of optimum interval separation possible statically)
- Time constraints specified via *checkpoints*
  - User-defined checkpoints are inserted in the source code and time constraints between checkpoints are defined.

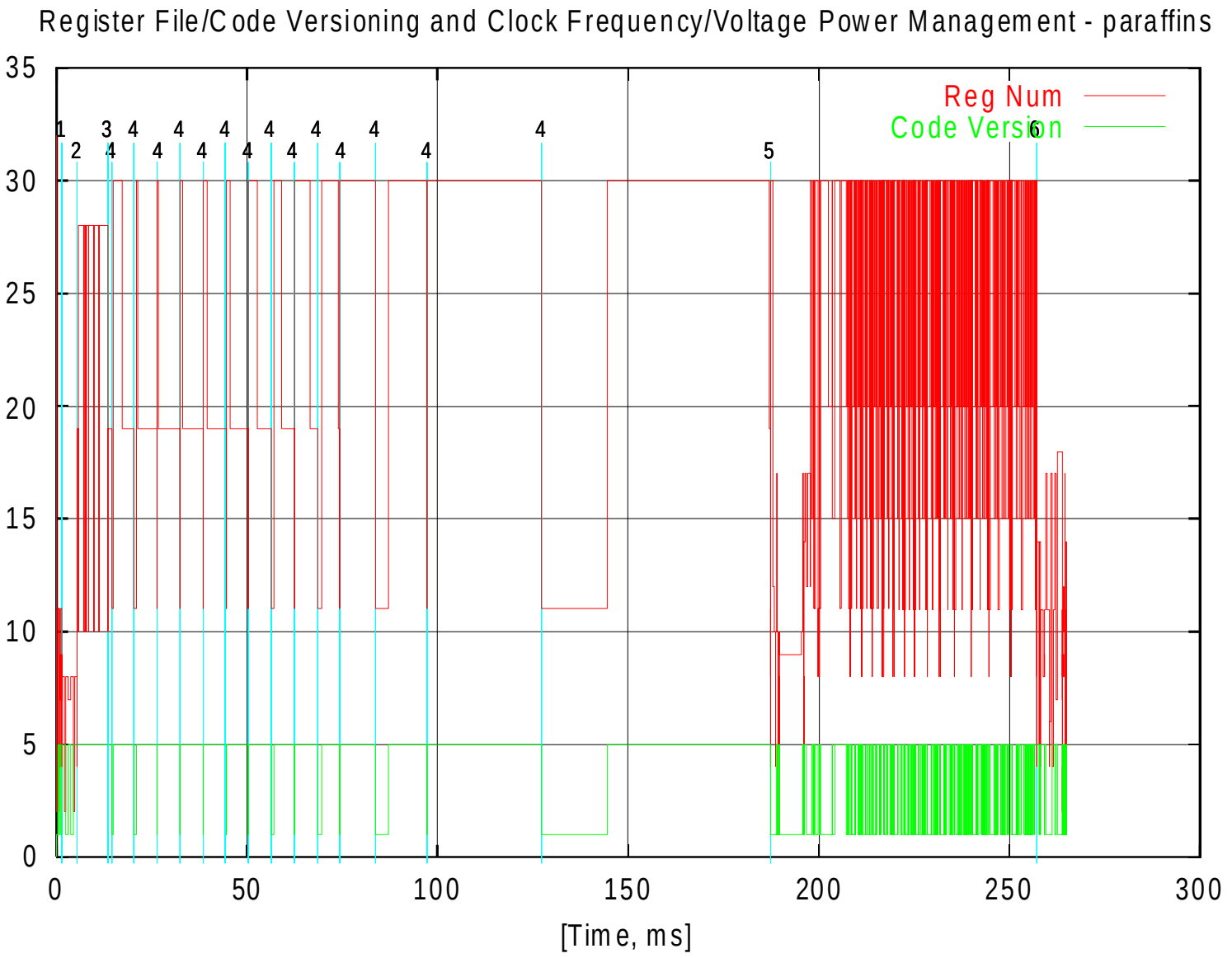
Constrained Dynamic Frequency & Voltage Scaling

- Specify time and energy constraints
  - energy constraints specified via estimation of the varying power available throughout the whole program execution
- Power-performance profiling compiler
  - estimates max energy/cycle ratio and cycle count between checkpoints
- Run-time scheduler
  - Calculates run-time freq limit based on available power and energy profile between curr chp and all possible next chps
  - Calculates optimal target freq based on both time constraints and run-time freq limit between curr chp and all possible next chps.
  - Final target freq is selected so that the code runs as slow as possible within the imposed time constraints.

Combining DRR & F/V Scaling: Three Phases

- **Profiling phase 1: DRR**
  - **Using code annotations for DRR, run the program adjusting register file size and collect energy and cycle count for each function for all code versions**
  - **Select best code version for each function based on profile and save this info as code annotations.**
- **Profiling phase 2: Checkpoint verification**
  - **Using code annotations from phase 1, run the program adjusting register file size and changing code version to be run, and collecting energy/cycle and cycle count between checkpoints**
- **Scheduling phase: use code annotations from P1, P2**
  - **At function calls, dynamically change code version and RF size**
  - **At program checkpoints and change points in the available power profile, dynamically adjust frequency and voltage (choose operating point on speed-power curve).**

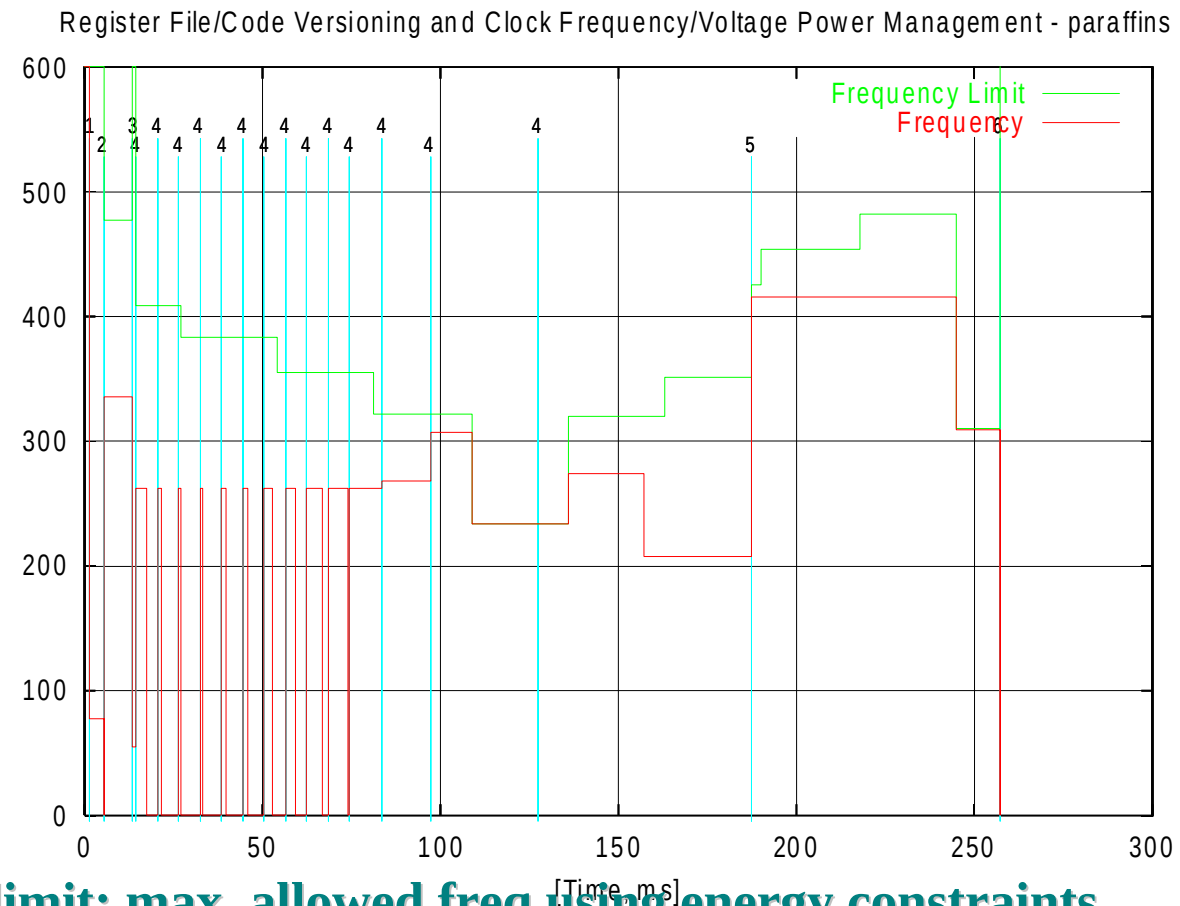
Register usage during execution



Results

#start #checkp	end checkp	MinTime (ns)	MaxTime (ns)
1	2	4000	4000
2	3	8000	8000
3	4	1000	1000
4	4	6000	30000
4	5	40000	60000
5	6	70000	70000

Time	Power
2000	1.5
27000	1.24
54000	0.98
81000	0.73
109000	0.28
136000	0.72
163000	0.95
190000	1.15
218000	1.38
245000	0.365
272000	0.85
300000	0.95



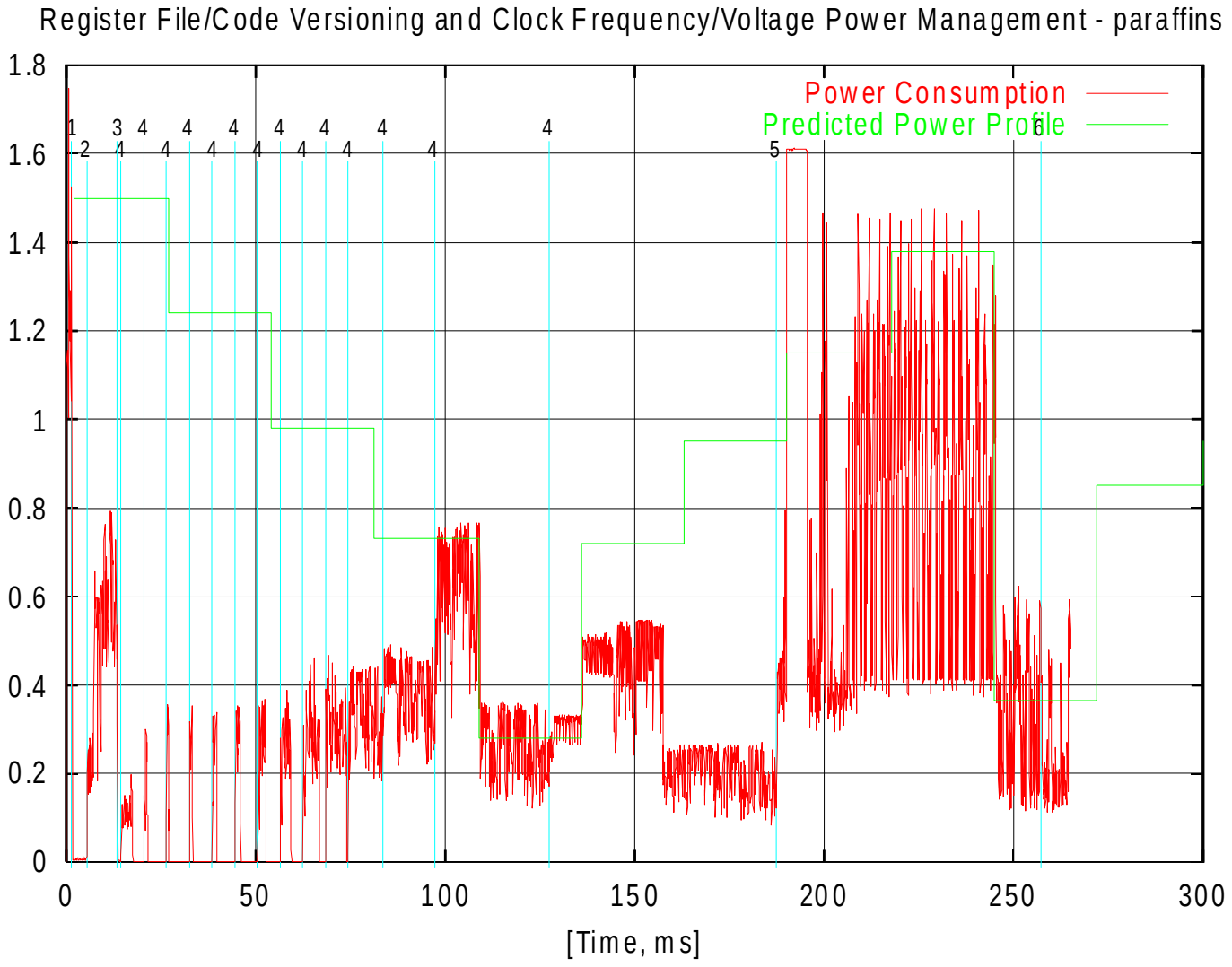
Freq limit: max. allowed freq using energy constraints

Target frequency chosen based on time and energy constraints

(Explanation)

- Green line
  - Maximum allowed freq calculated using energy constraints
- Blue lines
  - Program checkpoints
- Red line
  - Target freq chosen by dynamic scheduler, respecting time constraints and allowing the program run as slow as possible to save power
  - Freq value =0 means extra delay was inserted to satisfy minimum time constraints between checkpoints in the simulation

Combined Register Reconfiguration, F&V Scaling



Compiler Control of Power

- While average power reduction is important, effective control of dynamic power consumption is essential
  - especially for software management of power and performance
- The hard problem here is
  - identification of effective architectural mechanisms and their deterministic control through software
- COPPER approach
  - use architectural features common to a range of processor architectures
    - memory hierarchy, register files, instruction issue.
  - Coordinate with technology and OS strategies
    - frequency and voltage scaling.

Operating System Directed Power Management

OSPM

OnNow

ACPI

An API for Application-directed DPM

Operating System Directed Power Management

- Significant opportunities in power management lie with application-specific “knobs”
  - quality of service, timing criticality of various functions
- OS plays an important role in allocation, sharing of critical resource
  - it is a logical place for dynamic power management
  - application-specific constraints and opportunities for saving energy that can be known only at that level
- Needs of applications are driving force for OS power management functions & power-based API
  - collaboration between applications and the OS in setting “energy use policy”
    - OS helps resolve conflicts and promote cooperation

Example: Voltage Scheduling in General-purpose OSs

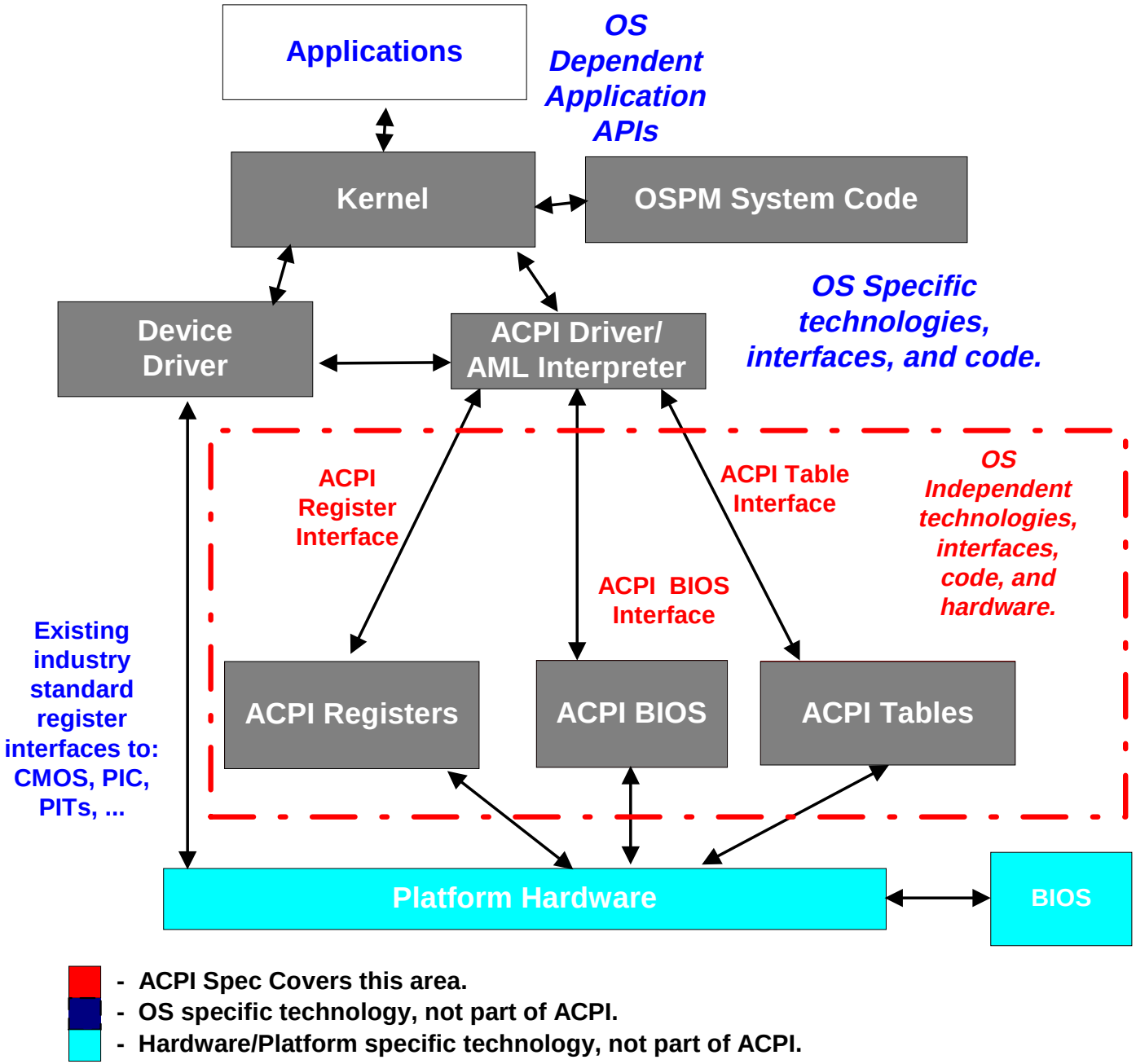
- Approach #1: [Weiser94]
  - time divided into 10-50 ms intervals
  - f & V raised or lowered at the beginning of the interval based on CPU utilization during the previous interval
    - 50% savings for a processor in the range 3.3V-5V
    - 70% savings for a processor in the range 2.2V-5V
- Approach #2: [Govil95]
  - predicts CPU cycles needed in the next interval
  - sets f & V accordingly
  - many prediction strategies: some did well, others not

Operating System Power Management (OSPM)

- Based on User preferences
  - Run in Performance mode or Quiet mode or Maximize Battery mode
- Supported by Microsoft's desktop operating systems via APM - Advanced Power Management
  - OS/BIOS co-operation
  - When OS goes to idle condition it performs an access to a register that causes an SMI#
  - SMI handler puts system into low power state
  - APM required OS to trust the system BIOS

Current OSPM - ACPI

- Advanced Configuration and Power Management Interface (ACPI)
  - OS visible (SCI-based) as opposed to OS invisible (SMI-based)
  - OS/drivers/BIOS are in sync regarding power states
- Standard way for the system to describe its device config. & power control h/w interface to the OS
  - register interface for common functions
    - system control events, processor power and clock control, thermal management, and resume handling
- Info on devices, resources, & control mechanisms
  - Description Tables, linked in a "table of tables"
  - description data for each device:
    - Power management capabilities and requirements
    - Methods for setting and getting the power state
    - Hardware resource settings
    - Methods for setting hardware resources
- Thermal Management


Applications
OS Dependent Application APIs
Kernel
OSPM System Code
Device Driver
ACPI Driver/ AML Interpreter
OS Specific technologies, interfaces, and code.
ACPI Register Interface
ACPI Table Interface
OS Independent technologies, interfaces, code, and hardware.
ACPI BIOS Interface
Existing industry standard register interfaces to: CMOS, PIC, PITs, ...
ACPI Registers
ACPI BIOS
ACPI Tables
Platform Hardware
BIOS
- ACPI Spec Covers this area.
- OS specific technology, not part of ACPI.
- Hardware/Platform specific technology, not part of ACPI.

ACPI is not hardware or software Spec

- ACPI is an interface spec comprising of both hardware and software elements
- Three runtime components of ACPI
  - ACPI system Description Tables
  - ACPI registers
  - ACPI System firmware

Global System State Definitions

- In ACPI specification Global system states are defined by six principal criteria:
  - Does application software run?
  - What is the latency from external events to application response?
  - What is the power consumption?
  - Is an OS reboot required to return to a working state?
  - Is it safe to disassemble the computer?
  - Can the state be entered and exited electronically?

ACPI Global System State Definition

Global State	Software runs	Latency	Power Consumption	OS restart required	Safe to disassemble computer	Exit state electronically
G0 Working	**Yes**	**0**	**Large**	**No**	**No**	**Yes**
G1 Sleeping	**No**	**>0**	**Smaller**	**No**	**No**	**Yes**
G2/S5 Softoff	**No**	**Long**	**Very near 0**	**Yes**	**No**	**Yes**
G3 Mechanical off	**No**	**Long**	**RTC battery**	**Yes**	**Yes**	**No**

ACPI Device Power States (D0-D3)

- Device power states are states of particular devices; as such, they are generally not visible to the user. For example, some devices may be in the Off state even though the system as a whole is in the Working state.
- Device states apply to any device on any bus. They are generally defined in terms of four principal criteria:
  - Power consumption. How much power the device uses.
  - Device context. How much of the context of the device is retained by the hardware. The OS is responsible for restoring any lost device context (this may be done by resetting the device).
  - Device driver. What the device driver must do to restore the device to full on.
  - Restore time. How long it takes to restore the device to full on.
- Not All Devices have all the device power states D0-D3

Summary of Device Power States

Device State	Power Consumption	Device Context Retained	Driver Restoration
D0 –fully on	As needed for operation	All	None
D1	D0 > D1> D2>D3	> D2	< D2
D2	D0>D1>D2>D3	< D1	> D1
D3 - off	0	None	Full Initialization and load

Device Power States (cont.)

- Many devices do not have all four power states defined.
- Devices may be capable of several different low-power modes,
- Device class specific specifications describe which of these power states are defined for a given type (class) of device and define the specific details of each power state for that device class

Sleeping State Definitions

- **Within G1 (Global sleep state) various sleeping states are possible:**
  - **S1 Sleeping State -- low wake latency sleeping state. In this state, no system context is lost (CPU or chip set) and hardware maintains all system context.**
  - **S2 Sleeping State -- low wake latency sleeping state. This state is similar to the S1 sleeping state except that the CPU and system cache context is lost (the OS is responsible for maintaining the caches and CPU context). Control starts from the processor's reset vector after the wake event.**
  - **S3 Sleeping State -- low wake latency sleeping state where all system context is lost except system memory. CPU, cache, and chip set context are lost in this state. Hardware maintains memory context and restores some CPU and L2 configuration context. Control starts from the processor's reset vector after the wake event.**
  - **S4 Sleeping State -- the lowest power, longest wake latency sleeping state supported by ACPI. In order to reduce power to a minimum, it is assumed that the hardware platform has powered off all devices. Platform context is maintained.**
  - **S5 Soft Off State -- similar to the S4 state except that the OS does not save any context. The system is in the "soft" off state and requires a complete boot when it wakes. Software uses a different state value to distinguish between the S5 state and the S4 state to allow for initial boot operations within the BIOS to distinguish whether or not the boot is going to wake from a saved memory image.**

Power State Definitions

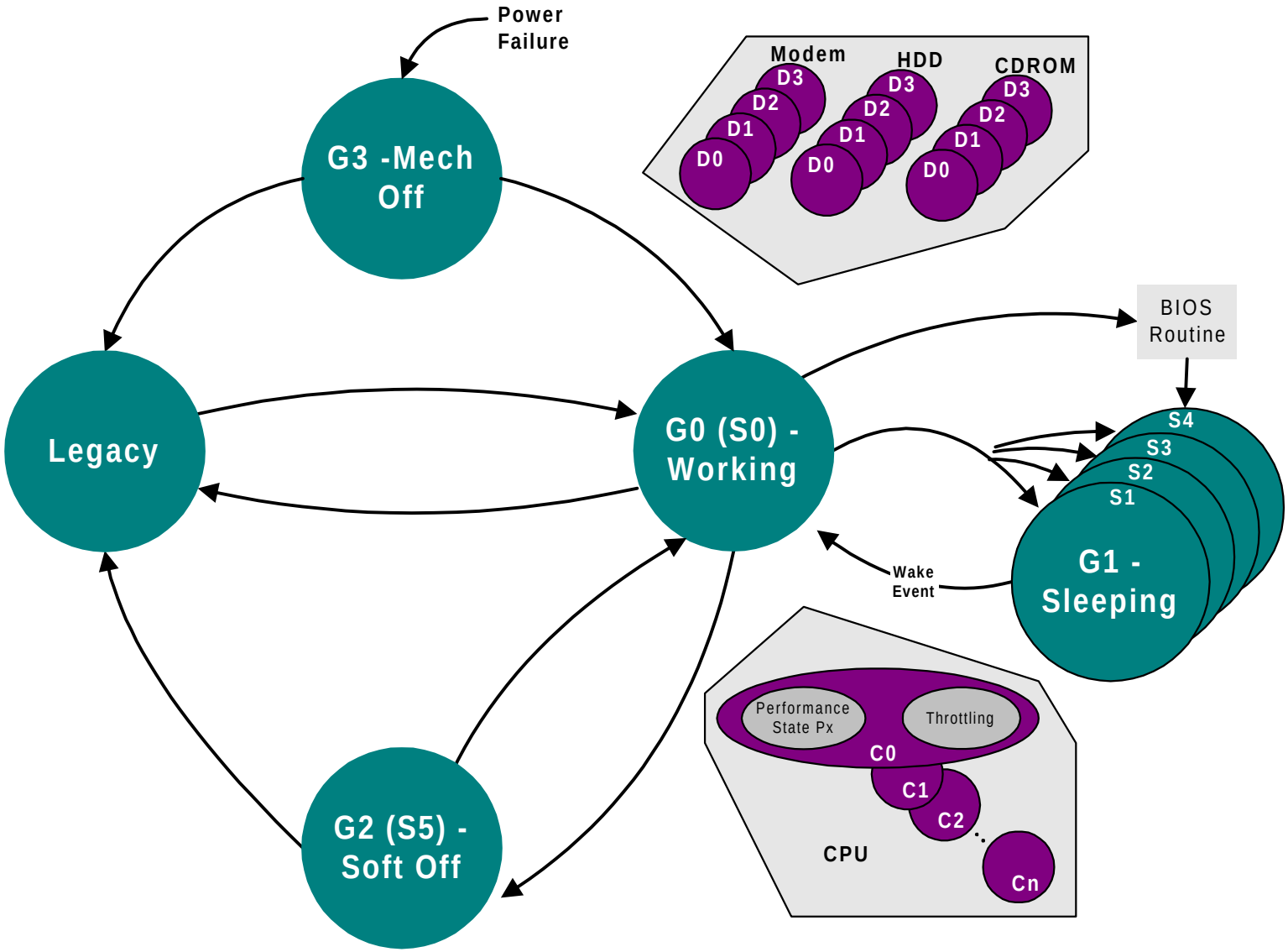
- **Within Global working state G0 various power states are possible:**
  - **C0 Processor Power State -- While the processor is in this state, it executes instructions.**
  - **C1 Processor Power State -- This processor power state has the lowest latency. The hardware latency in this state must be low enough that the operating software does not consider the latency aspect of the state when deciding whether to use it. Aside from putting the processor in a non-executing power state, this state has no other software-visible effects.**
  - **C2 Processor Power State -- The C2 state offers improved power savings over the C1 state. The worst-case hardware latency for this state is provided via the ACPI system firmware and the operating software can use this information to determine when the C1 state should be used instead of the C2 state. Aside from putting the processor in a non-executing power state, this state has no other software-visible effects.**
  - **C3 Processor Power State -- The C3 state offers improved power savings over the C1 and C2 states. The worst-case hardware latency for this state is provided via the ACPI system firmware and the operating software can use this information to determine when the C2 state should be used instead of the C3 state. While in the C3 state, the processor's caches maintain state but ignore any snoops. The operating software is responsible for ensuring that the caches maintain coherency.**

More on Device Power Management

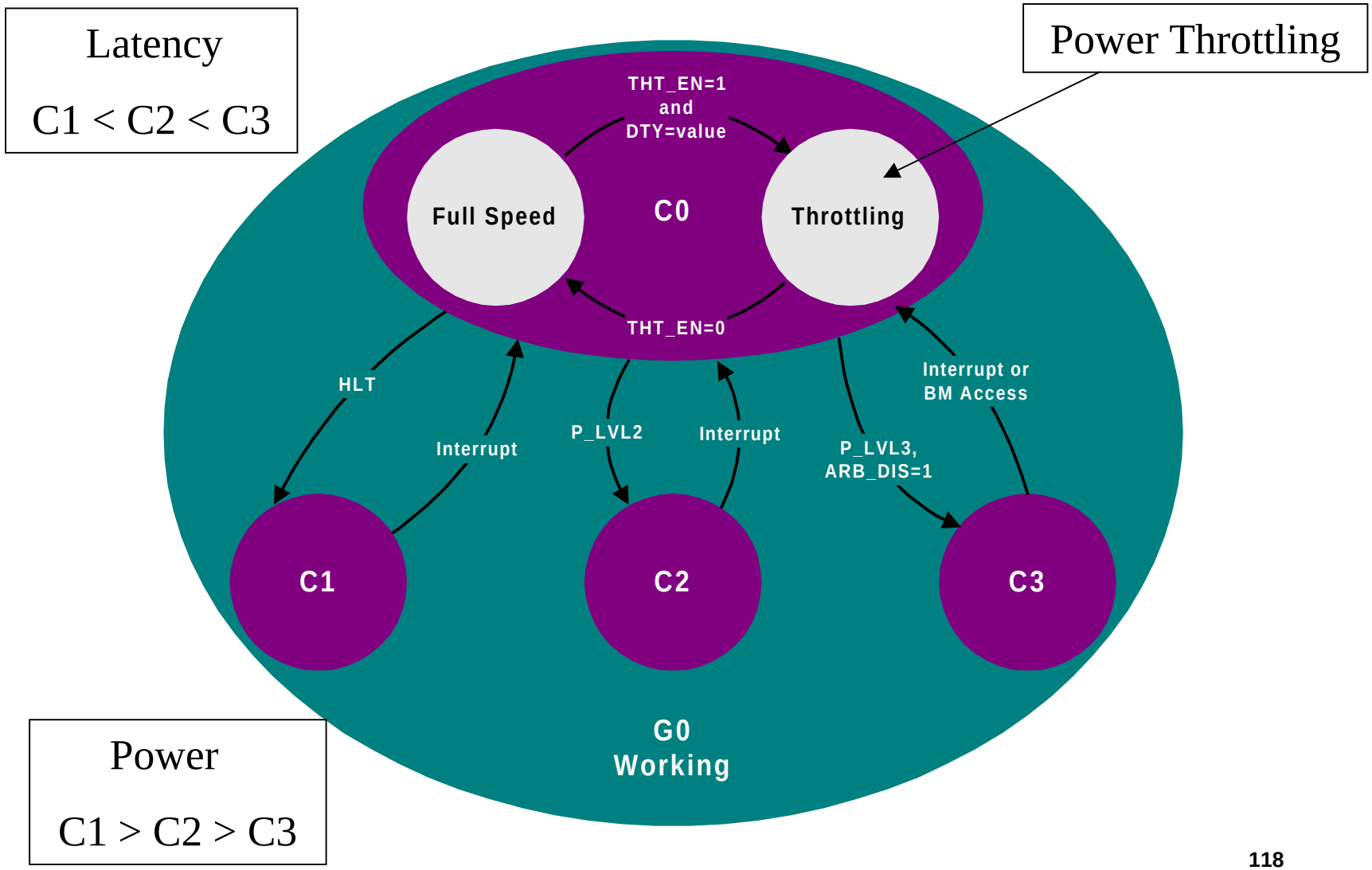
- The device power state definitions are device-independent
- However, the classes of devices on a bus must support some consistent set of power-related characteristics
- The follow class-specific power characteristics must also be standardized:
  - Device Power State Characteristics.
    - Each class of device has a standard definition of target power consumption levels, state-change latencies, and context loss.
  - Minimum Device Power Capabilities.
    - Each class of device has a minimum standard set of power capabilities.
  - Device Functional Characteristics.
    - Each class of device has a standard definition of what subset of device functionality or features is available in each power state (for example, the net card can receive, but cannot transmit; the sound card is fully functional except that the power amps are off, and so on).
  - Device Wakeup Characteristics.
    - Each class of device has a standard definition of its wake policy.
- ACPI provides the OS, the controls and information needed to perform device-specific power management.

Various Device Classes defined in ACPI standard for specialization of D0-D3

- Audio Device Class
- COM Port Device Class
- Display Device Class
  - CRT monitors, LCD panels, and video controllers
- Input Device Class
  - keyboards, keypads, mice, pointing devices, joysticks, and game pads, virtual reality devices
- Modem Device Class
  - modem and modem-like (for example, ISDN terminal adapters) devices
- Network Device Class
  - Ethernet and token ring adapters.
  - ATM and ISDN adapters are not supported
- PC Card Controller Device Class
- Storage Device Class
  - ATA hard disks, floppy disks, ATAPI and SCSI CD-ROMs, and the IDE channel.


Power
Failure
G3 -Mech
Off
Modem
HDD
CDROM
D3
D2
D1
D0
BIOS
Routine
Legacy
G0 (S0) -
Working
S4
S3
S2
S1
G1 -
Sleeping
Wake
Event
Performance
State Px
Throttling
C0
C1
C2
Cn
CPU
G2 (S5) -
Soft Off

ACPI Processor Power States



Overview of ACPI System States

State	CPU	Memory	Devices	Wake Up	Context Tracking
G0 Working	C0: Executing @ Full Speed C1:C3 Executing in PM state (ie Thermal Throttle/HLT)	Retained Power: ON Refresh: Normal	Powered Up & Down based on demand D0-D3		
S1 Sleeping	Not Executing Context Retained CPU CLK: OFF System CLK: ON Power: ON	Retained Power : ON Refresh : Normal	Devices Power down depending on wakeup & power requirements	Lowest Latency Restart @ CS:IP +1	H/W responsible for saving context of CPU, System I/O, & Memory
S2 Sleeping	Not Executing CPU/Sys Cache Context Lost CPU CLK: OFF System CLK: OFF Power: ON	Retained Power : ON Refresh : Standby / Auto	Devices Power down depending on wakeup & power requirements	Latency > S1 Restart @ Boot Vector	H/W responsible for saving context of System I/O & Memory OS responsible for saving CPU context
S3 Sleeping	Not Executing CPU/Cache Context Lost CPU CLK: OFF System CLK: OFF Power: OFF	Retained Power : ON Refresh : Standby / Auto	Devices Power down depending on wakeup & power requirements	Latency > S2 Restart @ Boot Vector	H/W responsible for saving Memory context BIOS restores Memory Controller Context. OS responsible for saving CPU & System I./O context
S4 S4BIOS Sleeping	Not Executing CPU/Cache Context Lost Everything: OFF	Context Lost Power : OFF Refresh : N/A	Devices Power down depending on wakeup & power requirements	Latency > S3 Restart @ Boot Vector	OS(S4) / BIOS(S4bios) is responsible for saving and restoring all system context, including memory
G2/S5 Soft OFF	OFF	OFF	Devices are OFF, Power Button Press will wake up the system	Latency > S4 Restart @ Boot Vector	OS uses S5 to turn the machine off

NOTES:
- OS chooses the lowest supported sleep state in which all enabled wakeup devices still functions under the latency requirements from apps.
- ASL binds each Sx state to a SLP_TYP value, which based on platform design of power planes & clocking logic det what portions of the h/w power down.
- For each Device, ASL lists which power resources are needed to maintain a ‘wakeup’ capable state
- ‘System I/O’ refers to Motherboard Devices: PIT, PIC, DMAC, NMI State....OS saves & restores this stuff for S3

Processor Power Management

- OS manages the power dissipation of the processor
  - OS chooses Cx state based on idle time above a threshold
    - Above threshold, OS uses lower power Cx state
  - ACPI Timer (use a hw timer from chipset) used for idle time detection
    - Timer sampled prior to and after exiting idle loop
  - Processor clock throttling
    - OS could scale processor duty cycle to match system usage

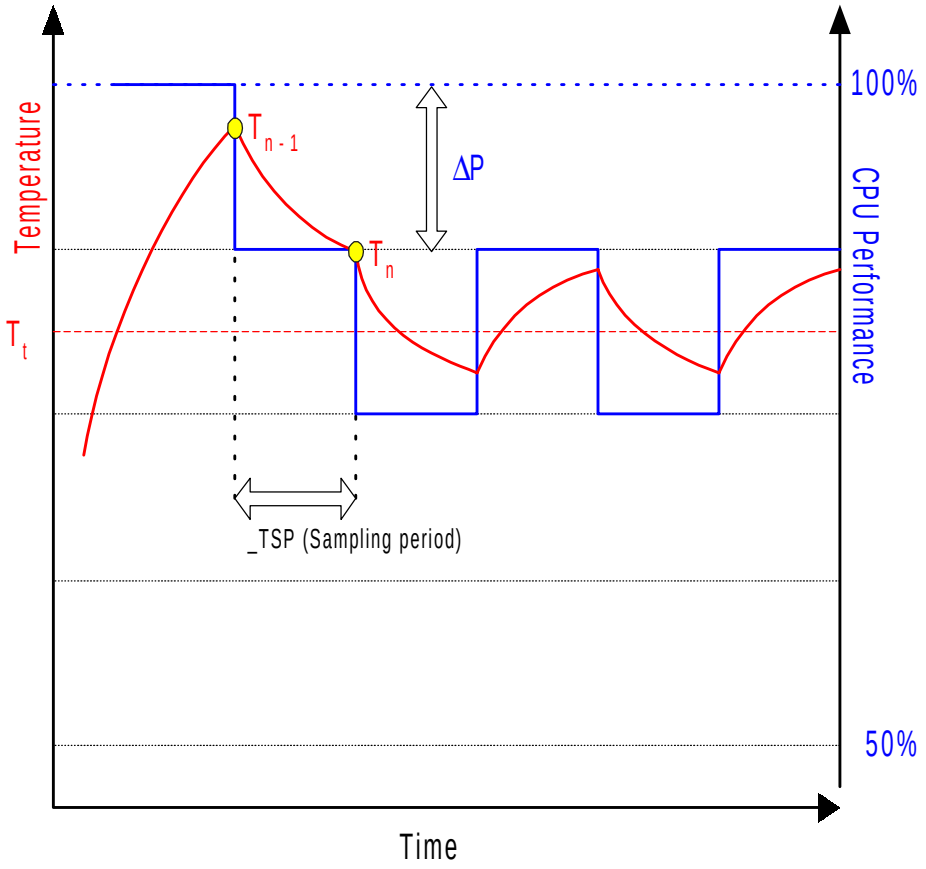
    E.g 25% idle time, processor performance throttled to 75%
    - Clock throttling has a longer latency today than C1 implementation
- OS uses power dissipation of processor to manage zone temperatures
  - ACPI allows thermal zones
    - Different thermal characteristics are allowed per zone
    - OS will use processor clock throttling to control temperature of the zone that includes the processor

ACPI Thermal Management Methods

- Active Cooling
  - Turn on/Speed up system fan when system is hot
  - Turn off/slow down fan when system is cool
- Passive cooling
  - Reduces processor power dissipation when system is hot
  - Restricts power by modulating processors STPCLK# pin
  - STPCLK# duty cycle roughly proportional to reduction in CPU thermal dissipation
- Separate trigger points for Active versus Passive

ACPI Thermal Model

- For passive cooling the OS actively monitors the temperature in order to cool the platform.
- The OS calculates the CPU performance change required to bring the temperature down
  - Predefined equation with OEM supplied constants
  - OEM defined sampling period



Source: Frank Binns, Intel

Response Time

- Operating system response times
  - SCI interrupt handler is NOT always the highest priority interrupt
  - SCI service routine potentially paged to disk
  - Results in a high latency between a thermal trigger point and OS induced response
    - Responses may vary from a small number of microseconds to several 10's of milliseconds

Summary of functional areas covered by ACPI

- System Power Management
  - ACPI defines mechanisms for putting the computer as a whole in and out of system sleeping states.
- Device Power Management
  - ACPI tables describe devices, their power states, the power planes the devices are connected to, and controls for putting devices into different power states.
- Processor power management
  - While the OS is idle but not sleeping, it will use commands described by ACPI to put processors in low-power states.
- Device and processor performance management
  - DPM to achieve desirable balance between performance and energy by transitioning devices and processors into different states when the system is active.

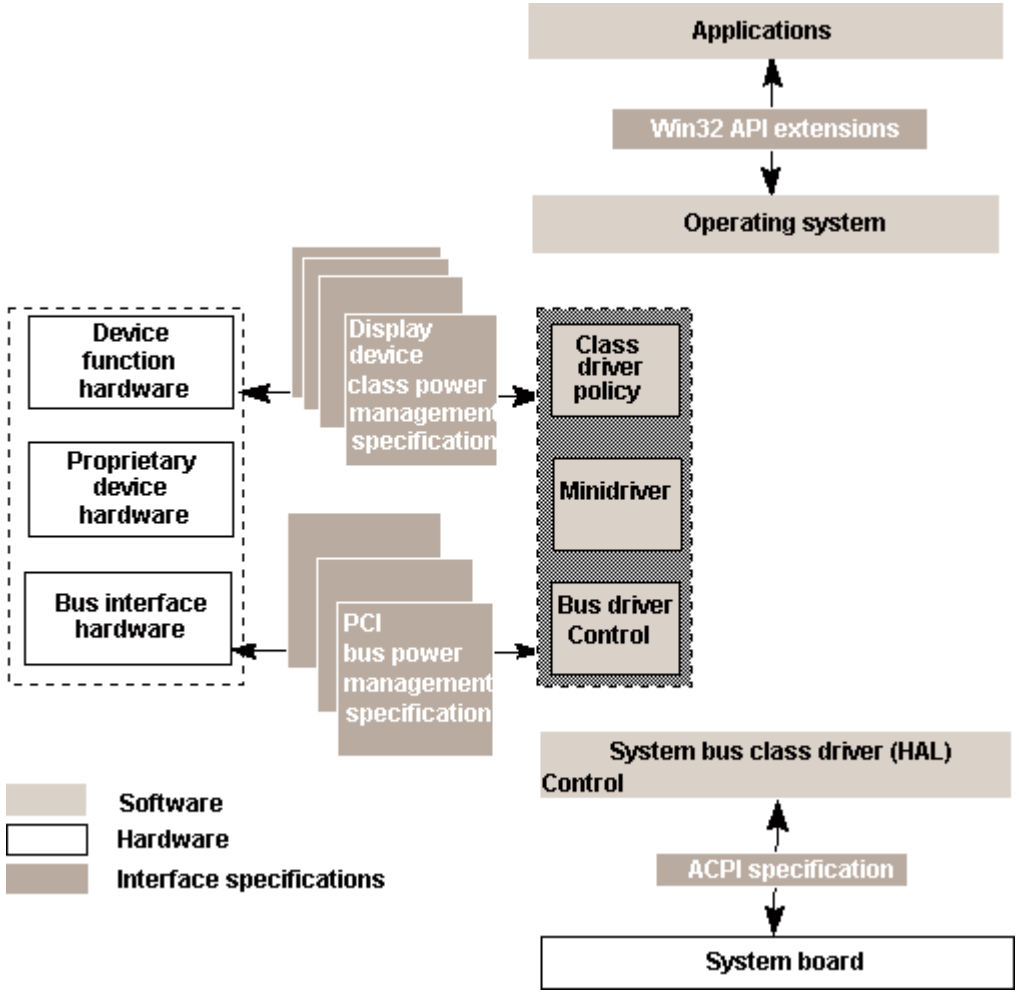
ACPI functionalities (cont.)

- Plug and Play
  - hierarchically arranged device and configuration information
- System Events
  - a general event mechanism for system events such as thermal events, power management events, docking, device insertion and removal, and so on
- Battery management
  - either through a Smart Battery subsystem interface controlled by the OS directly through the embedded controller interface, or a Control Method Battery interface.
- Thermal management
  - provides a model to allow OEMs to define thermal zones, thermal indicators, and methods for cooling thermal zones.
- A standard hw and sw interface between OS and Embedded Controller
  - allows any OS to provide a standard bus enumerator that can directly communicate with an embedded controller in the system, thus allowing other drivers within the system to communicate with and use the resources of system embedded controllers.

Microsoft's OnNow

- Win32 API extension allows applications to
  - affect the power management decision making
  - adapt to power state
    - find out if running on batteries so as to reduce processing
    - discover disk state & postpone low priority I/O
      e.g. paging
- Requires changes in hardware, firmware (BIOS), OS, and application software
  - bus & device power management standards for h/w
  - interface standard between OS & hardware
    - ACPI (Intel & Toshiba)
  - integration of power management into app control

OnNow Components

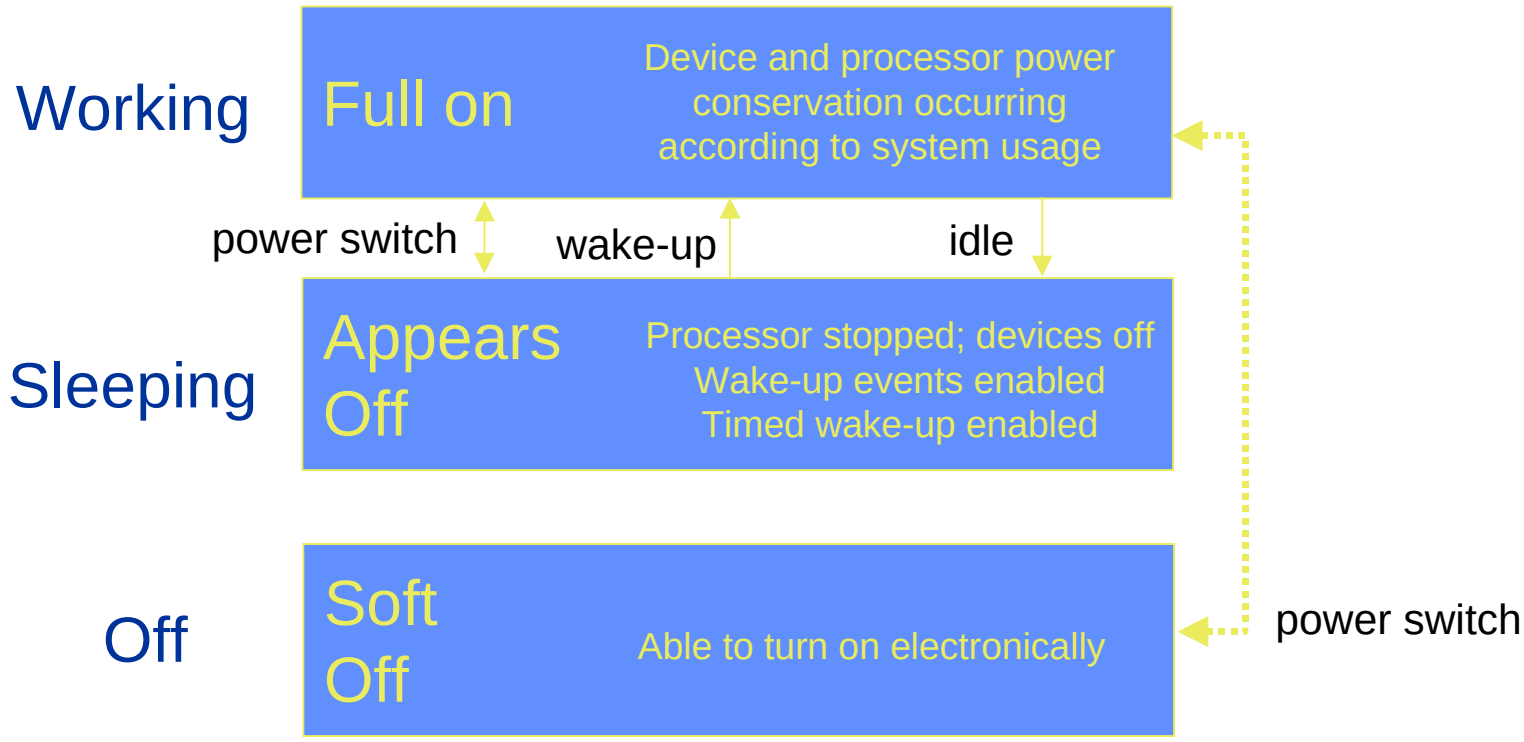


Ref.: Microsoft's "OnNow Power Management Architecture for Applications"

OnNow Architecture

- User's view: system is either on or off
- Reality: system transitions among a number of "power states" according to OS's power policy
- Global power states
  - working: apps are executing
  - sleep: software is not executing, & CPU is stopped
    - OS tracks user's activities & application execution states to decide when to enter sleep
      monitor user input, hints from applications
    - wake-up is time-based or device-based
  - off: system has shutdown and must reboot

OnNow Architecture (contd.)



OnNow Global Power States

Ref.: Microsoft's "OnNow Power Management Architecture for Applications"

OnNow Architecture (contd.)

- Power states for individual devices
  - managed by device drivers while system is “working”
    - function of application needs, device capabilities, and OS information
  - e.g. shutdown serial port if not in use
- Power states for CPU
  - OS transitions CPU between its various low-power states based on CPU usage
    - function of power source, processing time, user preferences etc.
- API mechanisms
  - for apps to learn about power events & status from OS
  - for apps to tell their requirements to the OS

Power Aware API for Efficient Power/Performance Management

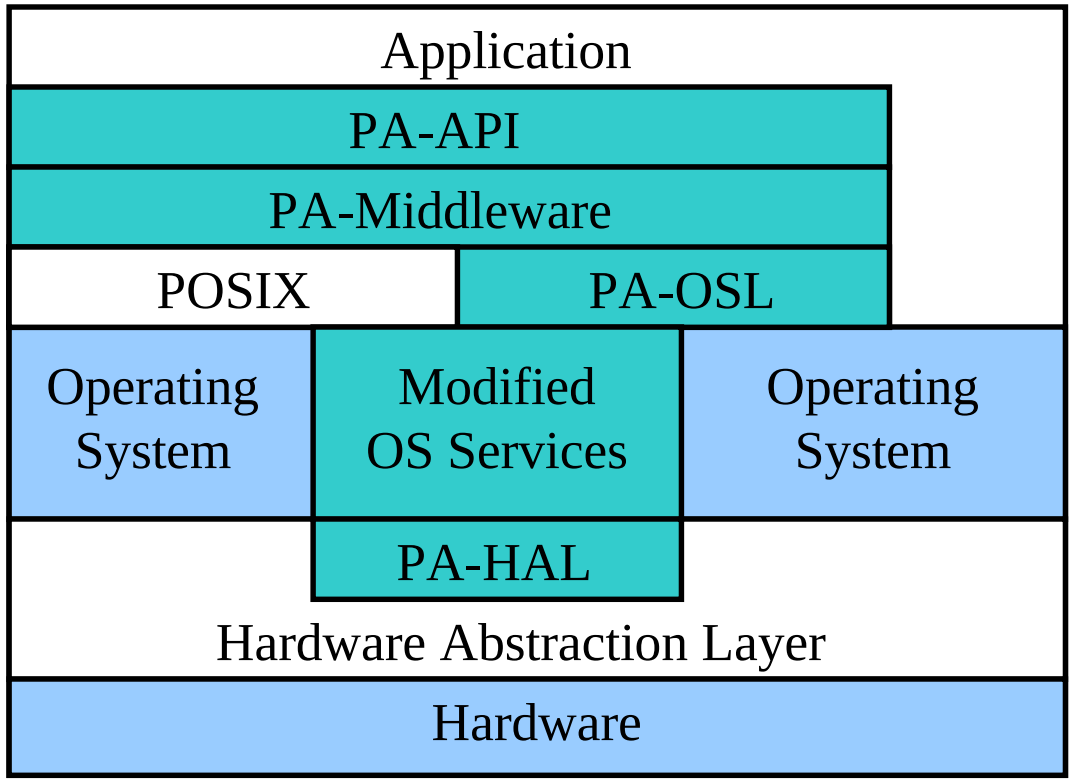
Enabling Application-OS Dialogues Under Real-Time Constraints

Goal

- Provide ways by which Application, Operating System and Hardware can exchange energy/power and performance related information efficiently.
- Facilitate the continuously dialogue / adaptation between OS / Applications.
- Facilitate the implementation of power aware OS services by providing a software interface to low power devices
  - A power-aware API to the end user that enables one to implement energy-efficient RTOS services and applications

Software Architecture

- **Consists on two-levels of software**
  - **between RTOS and underlying hardware**
  - **between Application and RTOS**



Software Architecture

- PA-API - Power aware function calls available to the application writer.
  - Some functions of this layer are specific to certain scheduling techniques.
- PA-Middleware - Power aware services
  - implemented on the top of the OS (power management threads, data handling, etc...).
- POSIX - Standard interface for OS system calls.
  - This isolates PA-API and PA-Middleware from OS.
- PA-OSL - Power aware OS layer.
  - Calls related to modified OS services should go through this level. Also isolates OS from PA-API and PA-Middleware.
- PA-HAL - Power Aware Hardware Abstraction Layer.
  - Isolates OS from underlying power aware hardware.
- Modified OS services
  - Implementation / modification of OS services in a power related fashion. Ex: scheduler, memory manager, I/O, etc.

Power-aware API Requirements

- Independent of Hardware and RTOS implementations
  - enables its use in different hardware platforms
    - for this all routines should access the HAL (Hardware Abstraction Layer) rather than the Hardware directly
  - enables its use in different RTOS as well as its use with different scheduling strategies
    - do not count on specific RTOS info and/or specific schedulers
- Services provided
  - processor frequency scaling and low-power state transitions
    - with costs of making such transitions
  - battery status (if the system is battery based)
  - appropriate routines to control energy-speed and energy-accuracy knobs available on I/O devices:
    - network interface, serial interface, LCD, etc.

Power-aware API

The applications interface provides the following services:

- The application is able to
  - tell RT information to OS (period, deadlines, WCET, hardness)
  - create new threads
  - tell OS time predicted to finish a given task instance
    - depending on the conditions of the environment (application dependent and not yet implemented)
- OS must be able to predict and tell applications the time estimated to finish the task
  - depends on the scheduling scheme used
- A hard task must be killed if its deadline is missed.

Current Status

- API specification available from
  - http://www.ics.uci.edu/~cpereira/pads/
- Implementation
  - eCOS RTOS:
    - open source, Object oriented and highly configurable RTOS (by means of scripting language)
  - Hardware platforms we are currently working with:
    - Linux-synthetic (emulation of eCos over Linux - debugging purposes only)
    - Compaq iPaq Pocket PC - StrongARM SA1110 based platform
    - Accelent IDP (Integrated Development Environment) - also StrongARM SA1110 based.
    - LRH Intel evaluation board 80200EVB - Intel Xscale based

DPM Algorithms Implemented

- A predictive RMS low-power scheduling
  - It validates the power-aware API implementation
    - assumes periodic tasks and ‘deadline = period’
  - The predictive scheduler implementation is divided as follows:
    - tables and variables manipulation
    - admission control and static slow down factor
    - dynamic slow down factor computation (time prediction)
    - deadline management (hard deadline tasks)
  - The processor frequency and voltage are scaled according to the time predicted by the OS
  - The application can also predict the execution time in order to enhance accuracy.

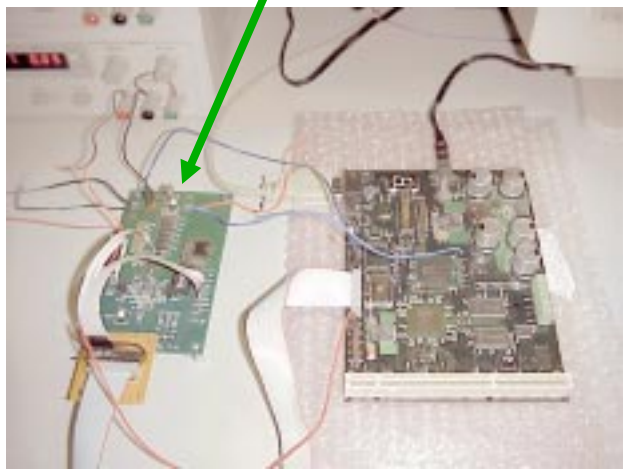
Implementation

- **eCOS Implementation:**
  - All the timing related information are kept internally to eCos kernel by means of tables
  - Some eCos classes were extended with new members in order to efficiently access the tables
  - The code is inserted in the eCos kernel source code by means of symbols definitions
    - enables automatic kernel synthesis of code
- **Plans:**
  - Extend eCos (by means of inheritance) class instead of just add code into them
  - Have a tool to generate the implementation of different scheduling schemes automatically (using the API)
  - API implementation on Embedded Linux

Implementation

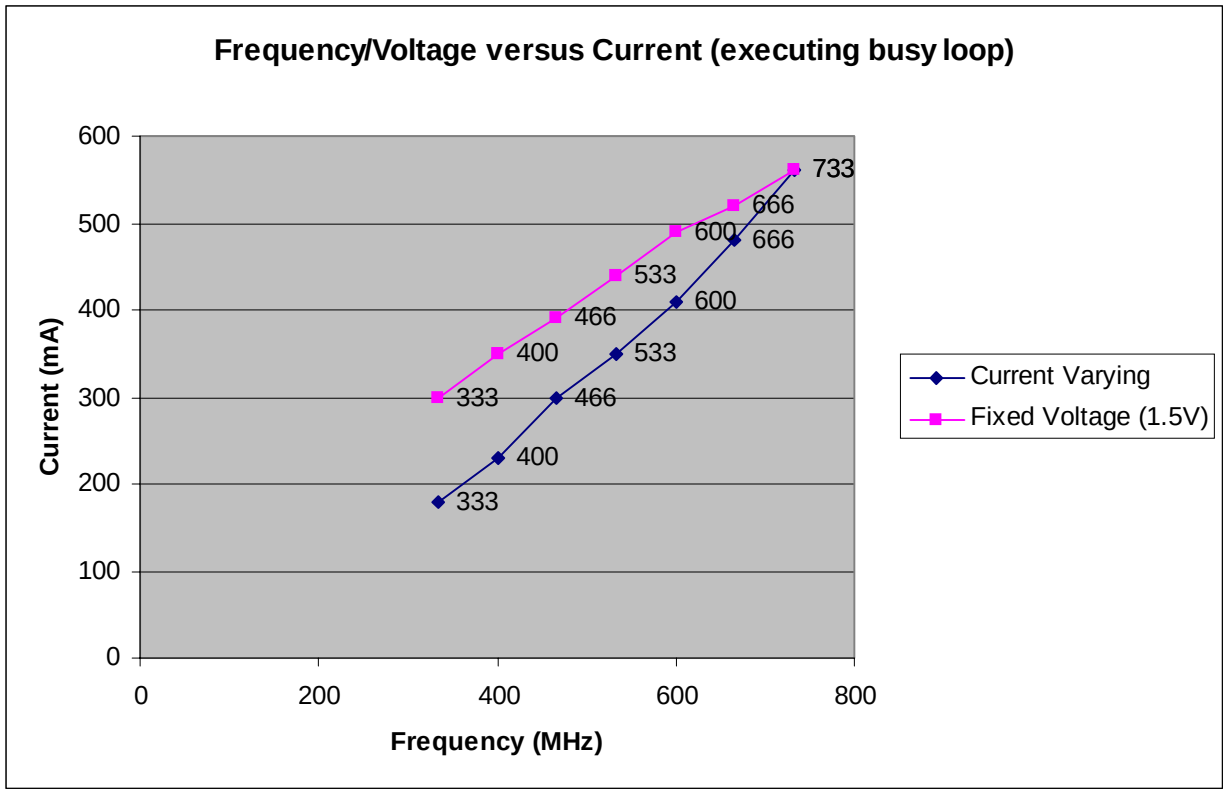
80200EVB w/ voltage scaling board and the host system

Maxim board for voltage scaling



Compaq IPAQ running eCos

Experiments - XScale Processor



For varying voltage

Frequency (Mhz)	Voltage (V)
333	1.0
400	1.1
466	1.2
533	1.25
600	1.3
666	1.4
733	1.5

*All measurements executing a busy loop

Outline

- Part I: Introduction to networked embedded systems
  - Energy consumption characteristics in NES
  - Power metrics
- Part II: Power management strategies
  - Effect of battery
  - Slowdown versus shutdown in DPM
  - Communications and networking strategies
  - Architectural, CPU and software strategies
  - OS strategies
- **Part III: Real Life Examples**
  - **Processors**
  - **Protocol standards**
- Summary

Intel SpeedStep

- Ramps up the processor clock when an AC source is connected
- Voltage reduced to 1.35V from 1.6V when in power saving mode, and clock reduced to 500 MHz (from 600-750 MHz)
- Support by BIOS, OS, chip set, and voltage regulator

Maximum Performance Mode			Battery Optimized Mode		
Frequency	Voltage	Max. Power Consumption	Frequency	Voltage	Max. Power Consumption
600 MHz*	1.35 V	14.4 W	500 MHz	1.10 V	8.1 W
600 MHz	1.6 V	20.0 W	500 MHz	1.35 V	12.2 W
650 MHz	1.6 V	21.5 W	500 MHz	1.35 V	12.2 W
700 MHz	1.6 V	23.0 W	550 MHz	1.35 V	13.2 W
750 MHz	1.6 V	24.6 W	550 MHz	1.35 V	13.2 W

*Low Voltage Version

AMD PowerNow

- Same as Intel's SpeedStep
  - used in 550MHz and 533MHz versions of AMD's Mobile AMD-K6®-2+ family of processors
    - core voltages of 1.4 to 2V
    - 0.18 micron technology
  - claim to extend battery life by up to 30%
- Three modes
  - high Performance mode - the CPU runs at maximum frequency and voltage.
  - battery Saver Mode - the CPU runs at lowest frequency and voltage to maximize system battery life.
  - automatic Mode - the system monitors application usage and continuously varies operating frequency and voltage to deliver performance on demand while optimizing battery life.

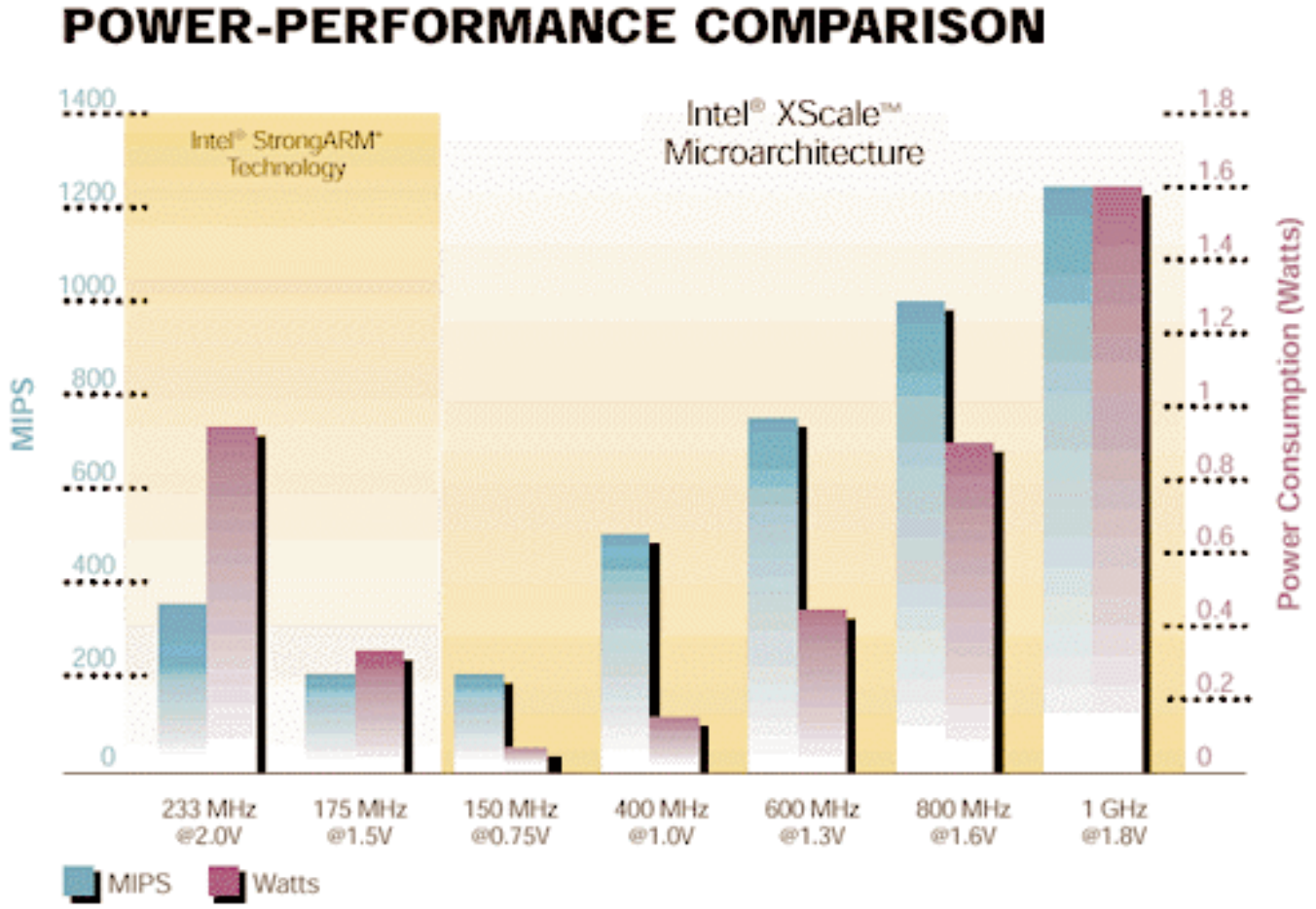
Power Aware Processors

- New processors in a different MIPS/Watt class with lots of "power-awareness" knobs
  - Intel Xscale or StrongARM-2 (0.18 micron)
    - 0.9 mW/MIPS (Dhrystone MIPS)
  - TI's TMS320C55 series low power DSPs
    - 0.05 mW/MIPS (TI's MIPS)
      - ➤16 mW at 320 MIPS /160 MHz
  - Transmeta Crusoe 5600 (0.18 micron)
    - 1.8x energy range: 1 W @ 1.2V/500MHz to 2W @ 1.6V/700MHz
  - Berkeley's lpARM (0.6 micron)
    - 9x energy range: 1.8 mW @ 1.1V/8MHz to 220 mW @ 3.3V/100MHz
- Variable voltage regulators
  - e.g. MAX1718 0.6V to 1.75V with 32 settings in ~ 126 μS

Intel's Xscale (StrongARM-2)

- Ultra-low power + high performance
  - via 7-stage pipeline ("Superpipeline")
  - Compliant with ARM 5TE ISA
    - 16-bit Thumb instructions, additional DSP instructions
- Power-awareness features
  - Dynamic voltage and frequency scaling on the fly
    - energy per op dynamic range of ~6x in SA-2 vs. ~2x for SA-1
      - ➤SA2: 1 mW (standby); 40-mW/185-MIPS @ 150-MHz/0.75-V ; 450-mW/750-MIPS @ 600-MHz/1.3-V; 900-mW/1000-MIPS @ 800-MHz/1.6-V
      - ➤SA1: 33-mW @ 59-MHz/0.8-V to 360-mW @ 206-MHz/1.5V in 11 discrete steps
    - 30 μS PLL-relock vs. 150 μS for SA-1
      - ➤regulator transition time may be bottleneck?
  - Idle, sleep, and quick wakeup modes
    - 100 μW drowsy state
    - Functional block powered up only when needed
- MAC coprocessor for signal processing

XScale vs. StrongARM



Transmeta's Crusoe Processors

- Two processors
  - TMS3200
    - up to 400 MHz
    - deep sleep idle mode with power as low as 20 mW
  - TMS5400
    - up to 700 MHz
    - deep sleep idle mode with power as low as 8 mW
- Low power features
  - "Code Morphing"
    - compact hardware engine surrounded by a software layer
    - processor emulates x86 with the software layer translating to the native VLIW format of the underlying processor
      - ➤power efficient as software eliminates lots of transistors (?)
      - ➤independence from x86 instruction set allowed power conscious implementation of the hardware engine (e.g. VLIW)
  - Higher integration: integrated "Northbridge"

Crusoe Processors (contd.)

- **LongRun Power Management in TM5400**
  - adjust power consumption by adjusting clock frequency and voltage on the fly
  - software continuously monitors the demands on the processor and dynamically pick just the right clock speed (and hence power consumption) needed to run the application
    - ➤extremely quick, without involved OS or BIOS operations
  - code morphing software adjusts the voltage on the fly
  - cubic reductions in power
    - e.g. an application needs 90% of the CPU
    - shutdown alone will save 10% power
    - LongRun saves almost 30% (0.9^3 = 0.73)

Performance of Crusoe Processors

- Two metrics
  - Workload Completion Rate

    WCR = Mobile Workload Completed / Time to Complete Workload
  - Workload Completion Efficiency
    - WCE = Mobile Workload Completed / Energy Consumed to Complete Workload

TM5400 (266-533 MHz) vs. Mobile PIII (500 MHz)

- WCR

	Mobile PIII	TM5400	Ratio
Load OS	60	55.2	0.92
Windows Desktop Idle	60	60	1
Office 2000	60	48	0.8
Web Browser	60	60	1
MP3 Playback	60	60	1
DVD Playback	60	60	1

TM5400 (266-533 MHz) vs. Mobile PIII (500 MHz)

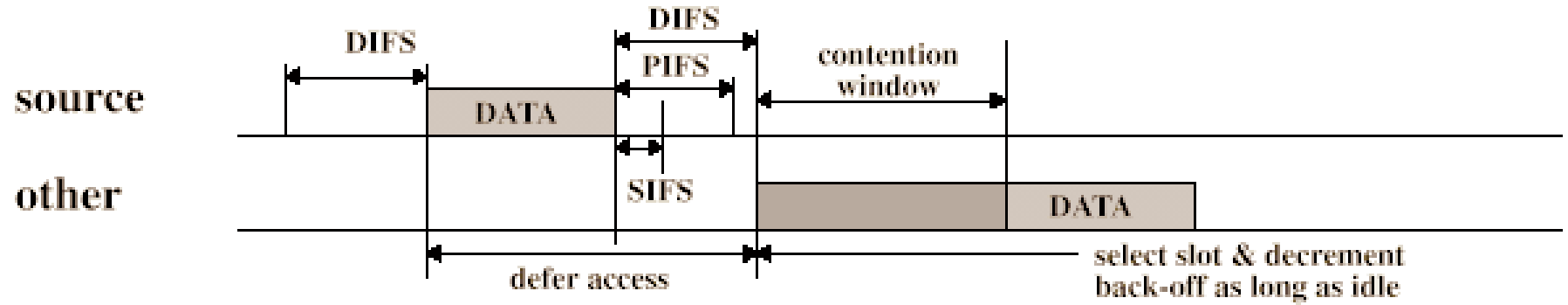
- WCE

	Mobile PIII	TM5400	Ratio
Load OS	5.85	20.0	3.42
Windows Desktop Idle	11.9	75.5	6.34
Office 2000	8.45	24.0	2.84
Web Browser	9.84	40.3	4.10
MP3 Playback	11.0	52.0	4.73
DVD Playback	8.85	27.6	3.12

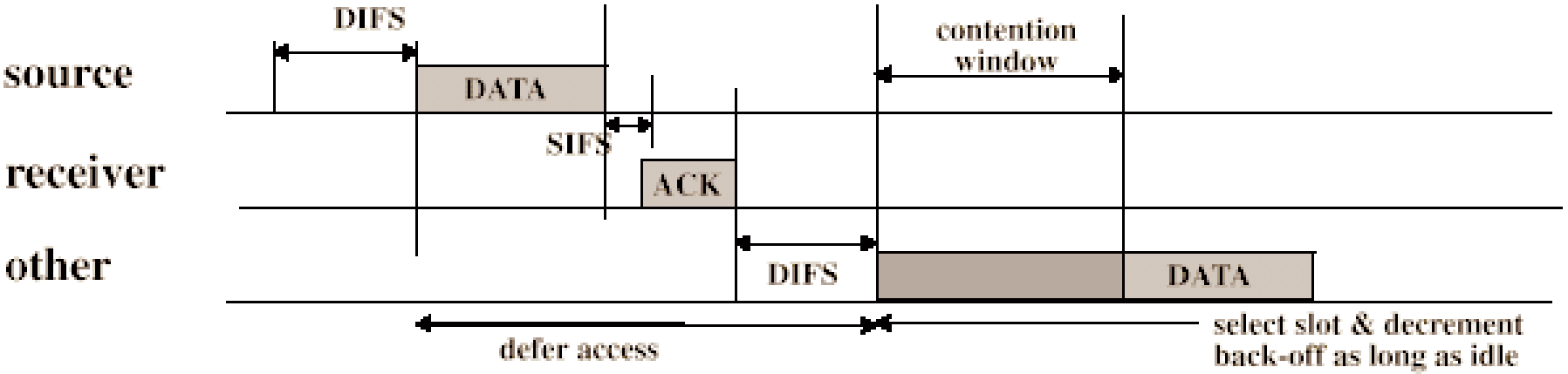
Power Management in 802.11 and BT

802.11 Review

- **CSMA/CA: direct access if medium free for > DIFS, else defer and back-off**
  - SIFS = short interframe space
  - PIFS = PCF interframe space
  - DIFS = DCF interframe space

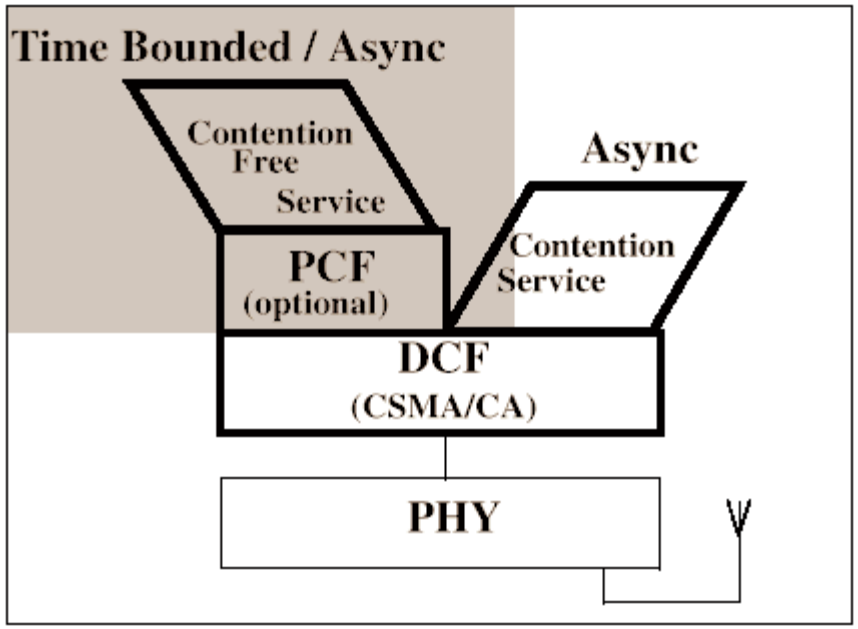


- **CSMA/CA + ACK: receiver sends ACK immediately if CRC okay**
  - if no ACK, retransmit after a random backoff

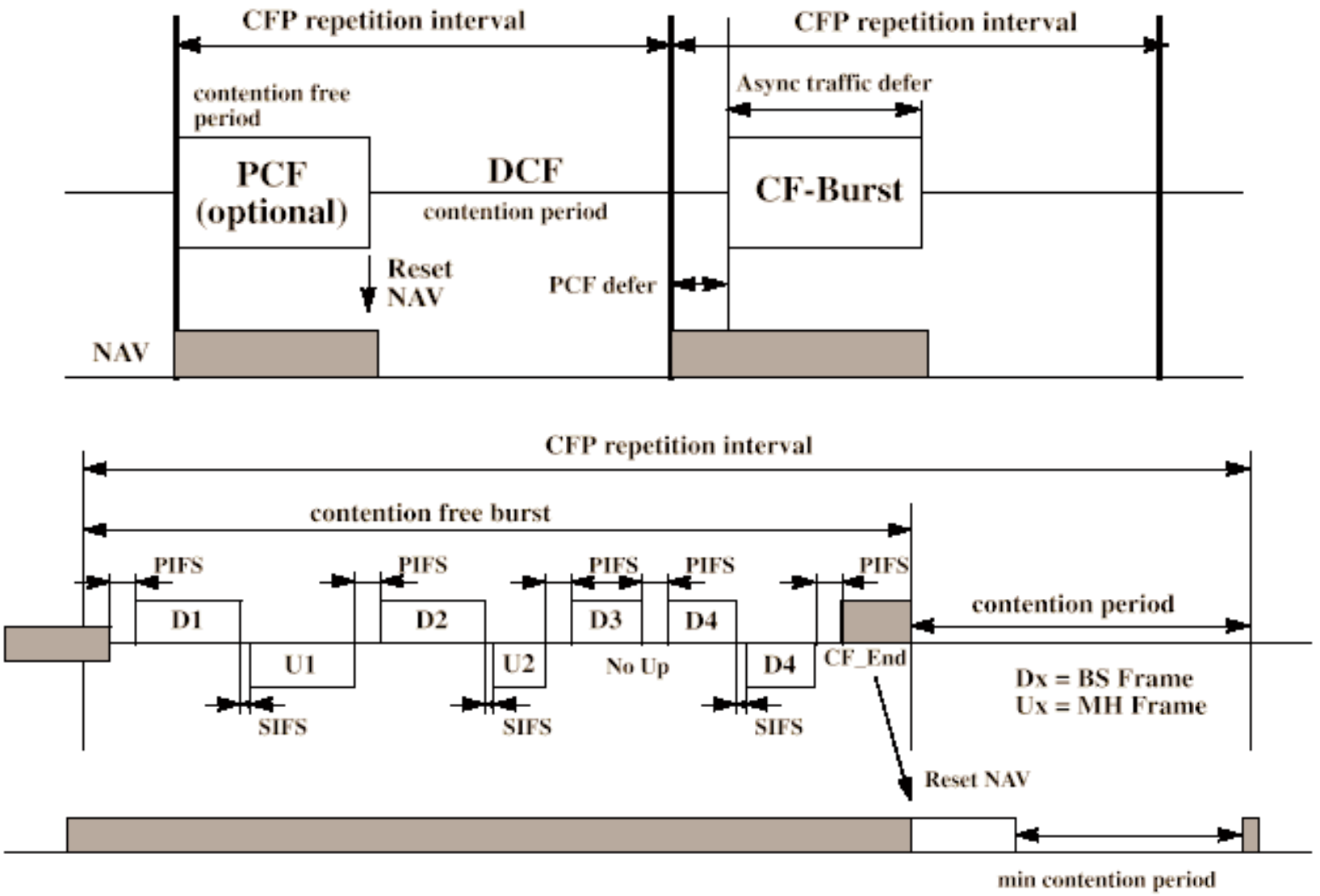


802.11

- Contention-free Point Coordination Function (PCF) to support low-jitter time bounded traffic
  - optional, resides at the AP



802.11 PCF Mode



Factors Affecting Power

- Cell size
- physical layer TX rate
- protocol overhead
- receiver sleep modes
- speed of wakeup
- efficiency of modulation (Eb/N0 for a given BER)
- antenna patterns
- acquisition aides
- system frequency uncertainty
- multipath abatement strategies

Typical 802.11 Strategies

- TX power control
  - capability to adjust transmitted power at the transmit IF to maintain a constant output power at the antenna port
    - transmit power reading from TX PA to the BB processor
    - allows TX PA to be driven close to its compression point without concern for overdriving it due to manufacturing variations (else backed off at least by 4 dB)
- single SAW filter for both TX and RX
- single oscillator for reference for RF/IF synthesizers, carrier timing and MAC clock
  - carrier and symbol timings are locks in new 802.11
- half duplex radio with unused portions turned off
- sleep modes for fast recovery
- low-power acquisition mode

Power States

- NIC Power down
  - all functions are powered down, no power
  - recovery time includes starting and initializing MAC, loading the initial values, starting the radio, synthesizer loading and acquisition, system slot time acquisition, channel scanning, beacon acquisition, system authentication
- Radio Deep Sleep
  - saves: initialization registers, beacon timing, system slot timing, last good channel, system authentication
  - requires MAC to be running while the radio is turned off
  - only synthesizers must require signal on power up
- NIC Sleep mode
  - MAC clocked with a low rate clock (32-KHz watch crystal)
  - saves most things as above except that slot timing is lost
- Radio Receive: all TX functions turned off
- Radio Transmit: all RX functions turned off

Power Saving in 802.11

- Mobile nodes in power saving (PS) mode switch off their radios for some period
  - sender nodes meanwhile buffer the frames
  - TX in a WLAN is active less than 2% of the time
  - Most of battery power is used by PHY RX circuitry
    - entire PHY can be turned off when no transfer is taking place even if the node (station) is active
  - Latency increase due to PS can be controlled by reducing timeouts in high layer protocols
- Nodes are synchronized to wake up at the same time when the sender announces buffered frames
  - nodes with frames for them in the announcement stay up until frame is delivered
  - timing synchronization function (TSF)
- Easy to do in PCF, but hard to do in DCF
- In PCF, the basic service set (a set of nodes on a logical network) can reduce consumption by 97.5% when cycling over one minute intervals over a continuously on system.

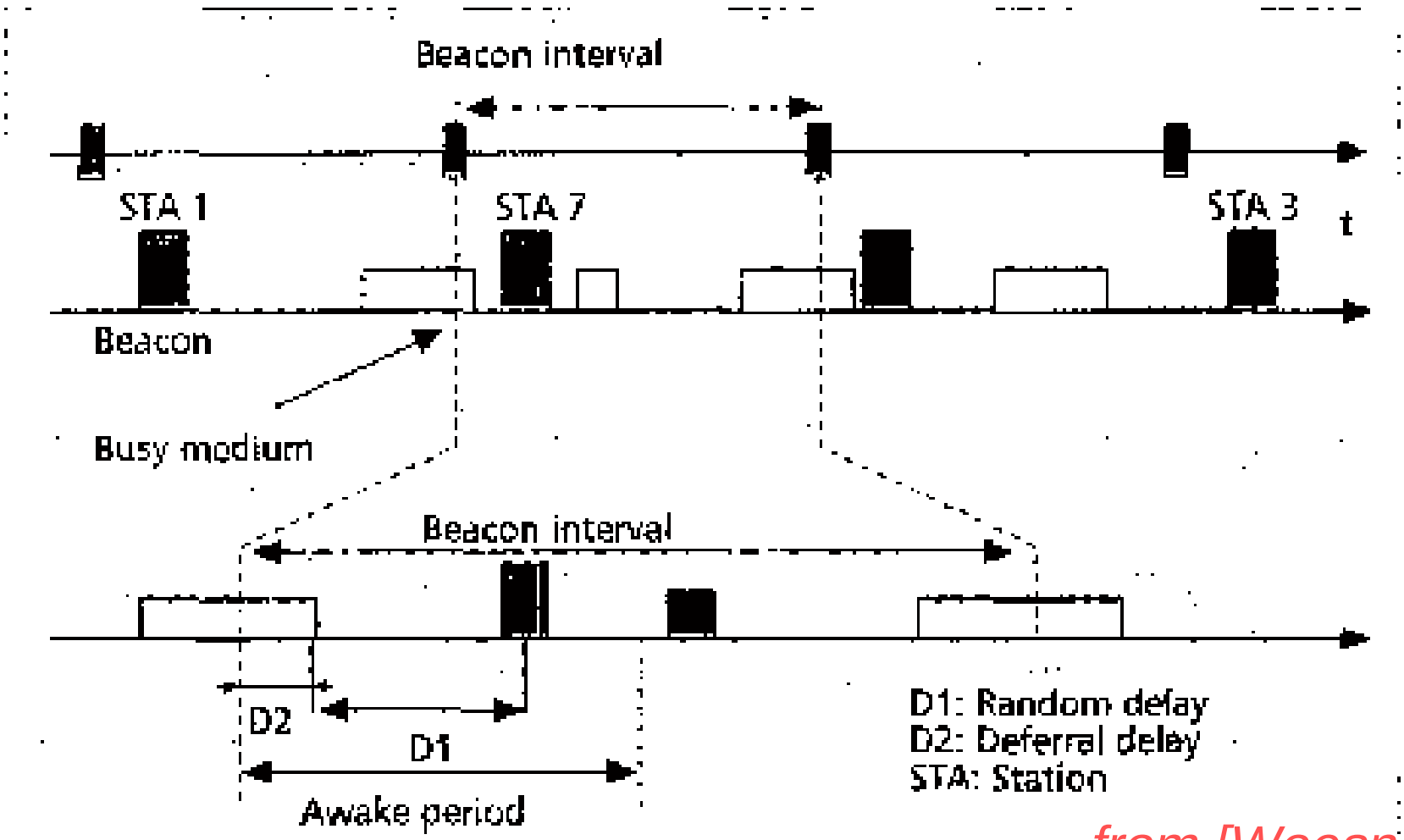
Power Saving in the PCF Mode

- AP generates time-stamped beacons and transmits them every beacon interval (~100 ms)
  - beacon transmission is deferred if channel busy
  - nodes wake up before the end of beacon interval and stay up until beacon is received
  - nodes adjust their local timers to the timestamp
- Beacon carries a traffic-indication map (TIM)
  - all unicast packets for nodes in sleep mode at announced in the TIM
  - mobile nodes with entries in TIM request packets from AP
- Broadcast packets are announced by a delivery TIM (DTIM) and send immediately afterwards

Power Saving in the DCF Mode

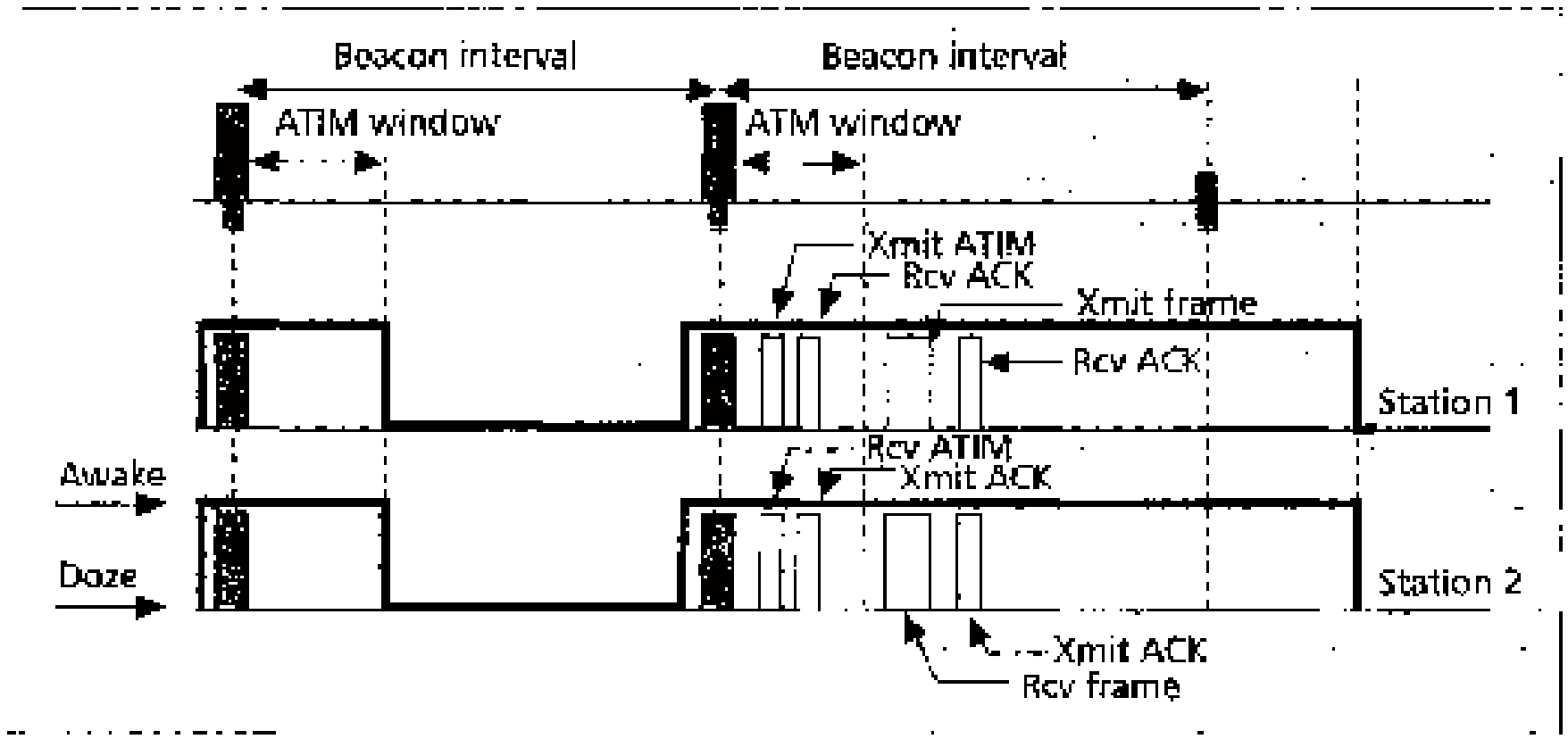
- Timers are adjusted in a distributed fashion
  - every node generates beacons
  - all nodes compete to transfer the beacon using DCF
  - the first node to transmit the beacon is the winner
  - other nodes cancel their beacons & adjust timers
- Packets for sleeping nodes are buffered by the sender until the end of beacon interval
  - announced using ad hoc TIMs (ATIMs) sent via DCF
    - transmitted in an ATIM window (~ 4 ms) after the beacon
    - ack'ed by the receiver
    - receiver stays up and waits for the packet

Power Saving in the DCF Mode (contd.)



from [Woesner98]

Power Saving in the DCF Mode (contd.)

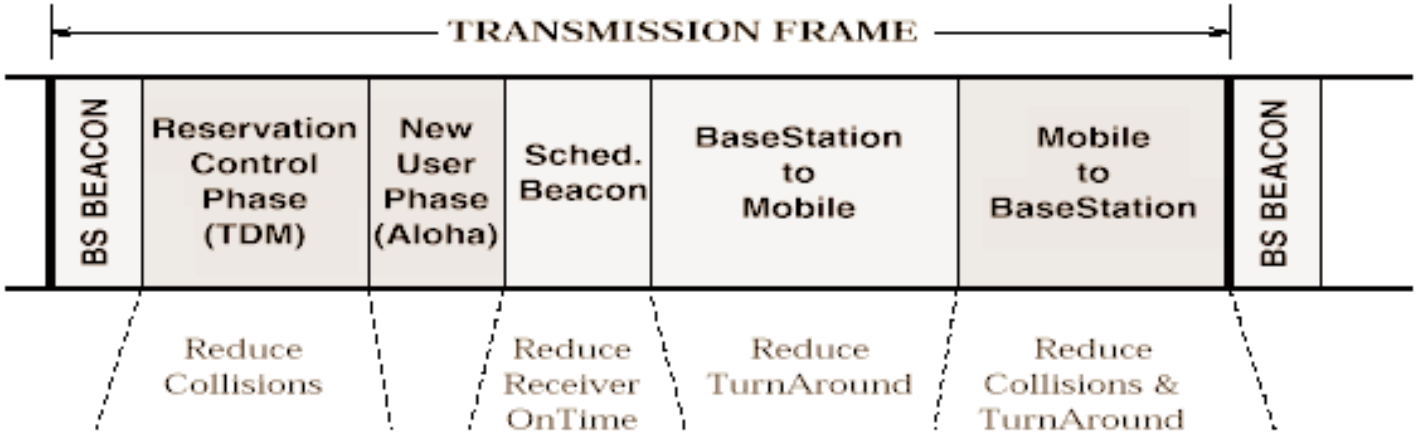


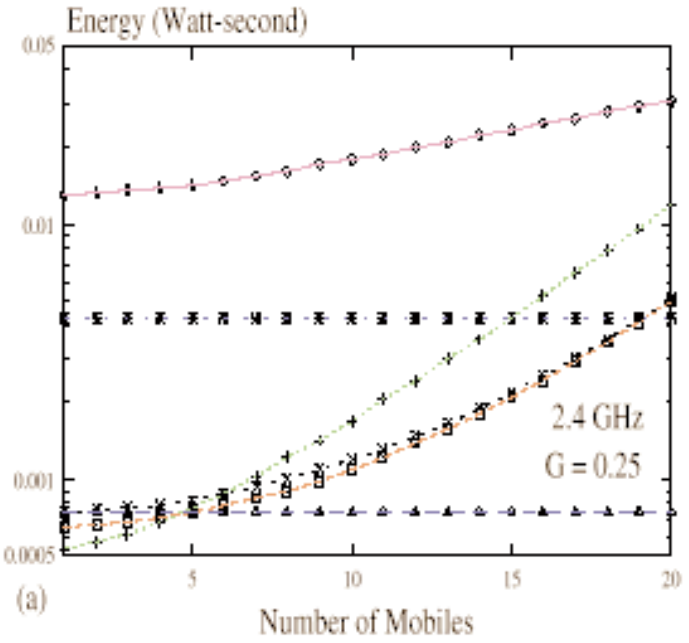
from [Woesner98]

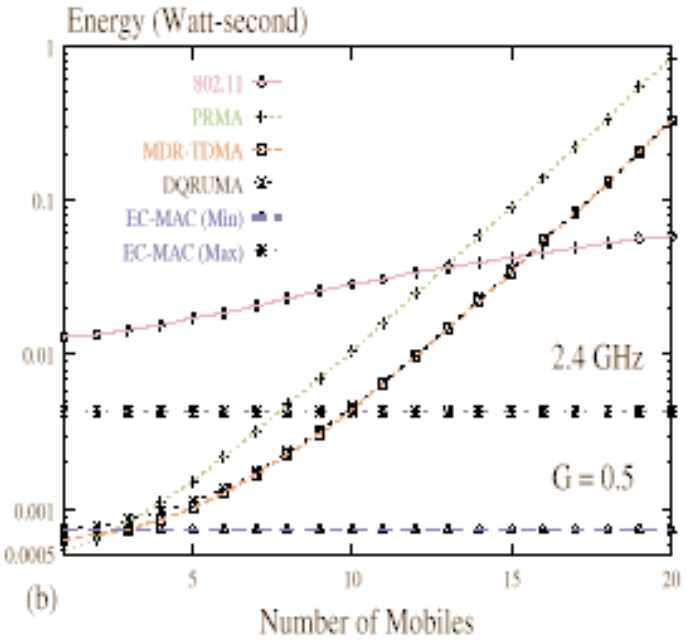
Power Saving in MAC via Directories or Schedules

- Scheduling access via a schedule or a directory
  - 802.11's TIM is like a directory
  - this concept also used in pagers
  - also suggested in the literature for application level
- Several TDMA-based MAC protocols around this idea
  - frame with directory or schedule carrying beacon in the first slot
  - mobile nodes wake up only in the right slots

EC-MAC: Energy Conserving MAC [Sivalingam97]





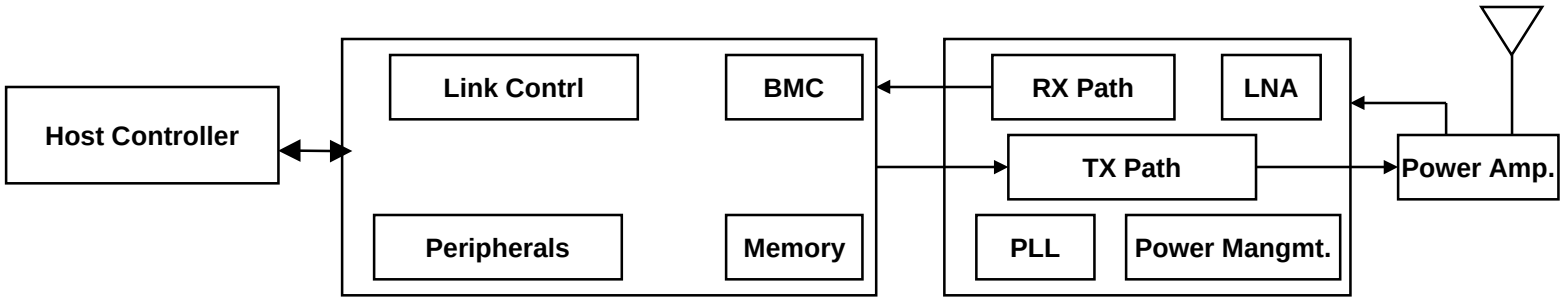


Energy Trade-offs at MAC Layer

- Careful control of access to minimize corruption due to interference
- Power management to minimize stand-by power
  - circuit techniques can help
  - even better: a wake-up channel
    - currently radios cannot know if and when another radio wants to talks to it
      - ➤periodic scheduled wake-up
    - wake-up signal using low power or passive components?

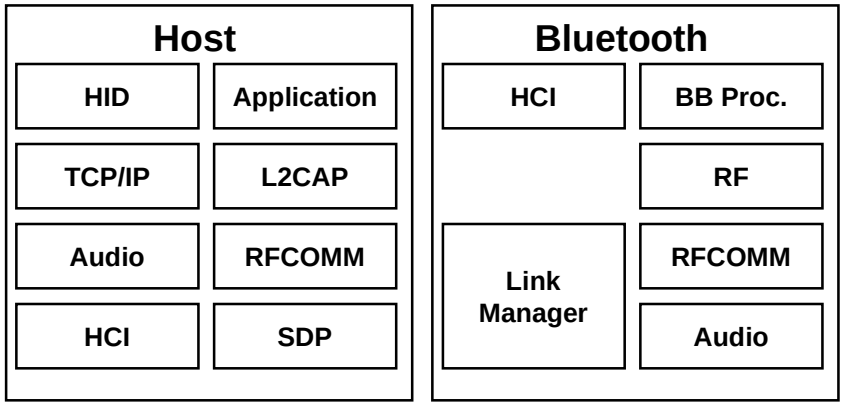
Optimizing Bluetooth Modules

- Power, size and cost tradeoffs for a range of applications
- A typical module includes:
  - Antenna
  - Power amplifier
  - RF section (2.4GHz ISM)
  - Baseband section (Link controller and manager)



  - Integration and optimization:
    - combine host and bluetooth processing
    - combine RF section and BB processing
    - integrate or eliminate memory system
    - eliminate power amplifier, PLL/VCO integration

Embedded SW in Bluetooth



- Embedded BT software upto link-layer host controller interface (HCI)
- Rest of the protocol stack can be optimized for size, power
- Baseband processing consumes about 15% of power
  - can be optimized further through sleep/shutdown.

BT Provisions for Low Power

- Robust hopping mechanism
  - master and slave remain synchronized even if no packets are exchanged for hundreds of ms (I.e., no dummy data exchange is needed)
- “Page mode” operation
  - receiver can quickly detect if a pakcet is present or not through a sliding correlator for an access code that lasts about 70 us after a scan of 100 us for jitter and drift.
- A master can put a slave in HOLD, PARK or SNIFF modes
  - control operation duty cycle while minimizing need for synchronization
- Duplexing through time division
  - no need for separate TX and RX oscillators
  - no need for a duplex filter
  - no cross-talk from TX into the RX path.

Summary

- "Computers with radios" present a very wide range of system optimization opportunities for power, size and performance
- Efficient power and energy management is key to enabling new range of applications
- Energy efficiency is a system level concern that cuts across subsystem components, functionality layers and its implementations
- Key components for power aware in wireless NES
  - low power IC components and their integration
  - energy efficient software implementation and compilation
  - dynamic power management that coordinates capabilities against application needs
  - energy efficient communications and networking
    - energy aware MAC, routing, transport
  - energy aware operating systems
- Applications programming need to be energy aware and provide "knobs" for the system designer to incorporate in DPM.